Manual de tradução de Jogos:
O fascinante mundo do ROMHacking

Pablo Muñoz Sánchez (conhecido como Pablito's)

Agradecimentos

Quero mostrar meus agradecimentos à todas aquelas pessoas que me ajudadaram durante o meu longo e duro cominho como romhacker. Sem elas, nem o presente manual nem minhas traduções poderiam ter visto a luz. Assim, desejo mostrar meus mais sinceros agradecimentos a Andrés Botero Castro(KaOSoFt) por ter revisado tão exaustivamente este manual com o propósito de que esteja livre de qualquer erro. Qualquer outra falha não será mais que responsabilidade minha. Por último, gostaria de dar o reconhecimento que merecem todas as pessoas que escreveram todos os documentos que recomendo ao longo deste manual e a todos aqueles programadores que têm facilitado tanto o trabalho do romhacker criando os programas utilizados no ROMHacking.

Correio eletrônico e página WEB

Pode entrar em contato comigo através deste e-mail: pms_sayans@hotmail.com

Visite minha página na web para mais informações: http://www.nekein.com/sayans/

Nota de abertura

A imagem usada foi pega de http://www.centenarioelquijote.com/ respeitando sua licença da Creative Commons.

Índice de Conteúdo

PRÓLOGO

Muito já aconteceu desde que, em meados do ano de 1997, um grupo de aficcionados tornaram público um patch cujo propósito era modificar uma quantidade considerável de bytes de um arquivo, de tal modo que, em vez de mostrar na tela alguns caracteres ilegíveis para muito desconhecedores do idioma japonês, mostrasse o texto perfeitamente em inglês. Tais aficcionados, sob o nome de RPGe, haviam traduzido o jogo Final Fantasy V do japonês para o inglês, depois de modificar exaustivamente os arquivos binários pertencentes a este jogo. Naquela época a emulação de consoles como o NES, Megadrive/Genesis e do SNES começava a fazer seus anseios e sonhos de desfrutar de jogos que nunca sairiam do Japão(e dos EUA no caso da Espanha) deixava pouco a pouco se conceber como tal. Assim, jogar Final Fantasy V deixava de ser uma fantasia graças ao aparecimento dos emuladores (muito distantes de serem perfeitos, isso sim), a barreira lingüística propunha um novo impedimento na hora de se divertir. O RPGe havia dado, um passo a mais ao permitir ao jogador compreender os indecifráveis diálogos japoneses. O ROMHacking havia nascido.

Este manual não pretende oferecer ao leitor uma quantidade enorme de informação desnecessária para traduzir um jogo. Portanto, segue com um ritmo rápido evitando um excesso de verbosidade, mas não passando por cima de detalhes relevantes no princípio que podem ser fundamentais em um projeto. Tão pouco tem como objetivo fazer uma retrospectiva de toda a história do ROMHacking, nem discutir sua legalidade, muito menos de teorizar uma realidade prática confundindo o leitor. Por último, já que por experiência própria sei que pode acontecer, o leitor não deve cair no erro de pensar que é inútil experimentar algo que possa falhar e que precisa de tempo, assim como ignorar aquilo que já sabe. É por isso que lhe digo agora que, ainda que frustração e desânimo nos invadam, e ainda que nunca consiga um resultado satisfatório, sempre haverá aprendido algo que, ainda que passem os anos, estará sempre na memória. Simplesmente precisará dos estímulos apropriados para invocar tudo o que foi aprendido.

Espero que com êxito, e sem deixar nada no tinteiro, passar tudo o que aprendi durante todos esses anos no mundo do ROMHacking. Muitos programas que facilitam o trabalho aparecerão desde a primeira versão deste tutorial, assim como uma infinidade de outros tutoriais que nos tem sido frutos da experiência. Não há o que duvidar de que é este manual que nos dará a clave da vitória diante de um caminho que reúne dificuldades dadas em outros vídeo-games.

Prólogo

Espero que meu tutorial sirva de algo na emuscene e que, acima de tudo, anime muita gente a traduzir seus jogos prediletos neste momento em que está escasso os “jovens” talentos. Sem mais o que dizer, daremos início por fim a este manual.

Pablo Muñoz Sánchez

Almería (Espanha), 4 de Julho de 2005

CAPÍTULO 1:
ASPECTOS TRADUTOLÓGICOS

Há de se supor que quem vá traduzir um jogo deva ter o desejo de vê-lo traduzido. Que seja você o autor da tradução, debe haver uma razão a se levar em conta, se bem que o ideal é que traduza um jogo para você mesmo e depois para os demais. É freqüente cair no erro de pensar que você possui a necessidade de traduzir para alguém quando, ao tratar-se de algo altruísta, não há quem deva sentir maior satisfação do fruto colhido do que você mesmo. E para isto, algo é preciso: Qualidade.

1.1. Requisitos para traduzir

Ainda que tratar isso tudo como uma afeição, e embora nada deveria reprovar isso mesmo a princípio, devemos levar em conta que é preciso tentar "reexpressar" o sentido original no texto em português, não só "traduzir". O que quero dizer com isso? Que ainda que conheça perfeitamente o idioma estrangeiro do qual se vai traduzir (algo que duvido, pois nem mesmo um nativo conhece totalmente o seu idioma), é preciso ter cuidado ao escrever o novo texto em nosso idioma. E não só com ortografia como muita gente pode pensar (é horrível ver erro em qualquer texto, mesmo que pareça trivial) senão pela expressão. Sou de opinião que só se deve traduzir para a língua materna, porque erros sérios de expressões nesta se consideram inaceitáveis.

O primeiro requisito para se traduzir não é saber um idioma estrangeiro, senão conhecer a língua do texto meta. Até mesmo sem ter nem idéia de chinês se pode escrever um texto em nossa língua materna que tenha sentido. Outra coisa é que seja fiel ao original. Obviamente, conhecer a língua de origem e sua cultura é também um princípio básico para a tradução, embora sempre se possa consultar um dicionário e outros recursos para nos ajudar, o que tem seus inconvenientes como veremos mais tarde. Para adquirir um alto nível de competência tanto em sua língua de origem como na língua meta, recomendo encarecidamente algo que esta fazendo agora mesmo: ler. Não só aumentará sua bagagem cultural (indispensável para traduzir), mas com o tempo aprenderá centenas de expressões e palavras que certamente lhe pareciam estranhas em dado momento. Lembro que há muitos anos, quando traduzia o Secret of Mana, a expressão "what on earth" me chocava muito. Seu sentido literal pouco tem a ver com "mas que diabos", seu equivalente em português (entre outras expressões similares).

1.2. O papel do contexto

A situação em que se dá lugar a um ato comunicativo, quer dizer, o contexto, é fundamental para compreender completamente qualquer palavra do enunciado. Conversando como um amigo, e com ânimo de oferecer um exemplo mais claro, proponho explicar a importância do contexto para compreender uma mensagem com a seguinte questão: dizer "Deus" quando alguém se surpreende ao observar uma proeza é o mesmo que dizer "Deus" em um momento de prazer? Eu creio que não. Pois que "God" em inglês corresponde a ambos os casos como o "Deus" em português. Mas isso nem sempre acontece assim.

É por isso que antes de traduzir um jogo é preferível jogá-lo antes e, ainda mais importante, traduzir enquanto se joga (uma prática provavelmente mais importante). Eu mesmo não traduzo sem antes jogá-los, e conforme traduzia me inteirava de coisas que poderiam fazer variar minha tradução, sobre tudo o registro dos personagens (digo, sua forma de falar). O exemplo mais claro se encontra no Final Fantasy IV, já que mudei muitas coisas enquanto revisava a tradução. Sem impedimentos, no Phantasy Star IV, como conhecia com perfeição toda a história e a personalidade dos personagens por tratar-se de um de meus jogos prediletos, não me surgiram muitas dúvidas na hora de dar um certo "toque" aos diálogos de Alys e Raja (dois dos mais carismáticos de vários protagonistas do jogo).

Por outro lado, vale lembrar que ler o texto de um jogo é muito diferente de ler uma novela. Um jogo, é igual a um filme, é caracterizado por possuir imagens e sons. Se um personagem falar "it's hot, isn't it? [Está quente, não está?]" quando tal indivíduo se encontra na neve, produzirá em alguns jogadores ao menos uma leve risada. Este caso não parece estabelecer nenhum problema de tradução (sempre e quando não tenha um duplo sentido), mas em determinados caso as imagens do jogo possuem um importância fundamental na hora de criar jogos de palavras que só darão dores de cabeça ao tradutor. É nestes casos que a criatividade do tradutor cobra sua verdadeira capacidade.

1.3. A invisibilidade da tradução

Uma tradução de qualidade será aquela que precisamente não se deixe notar, quer dizer, aquela que não pareça uma tradução. Citamos muito, nossos personagens de cinema favoritos, mas por que nunca pensamos que tais falas estão escritas iguais ao script? Acaso pensamos que "*mosquis"* ou "multiplicar por zero" são resultado de uma tradução? Isso é o que devemos pensar, que nossa tradução não se manifeste em nossos consumidores como tal. E para isso o requisito básico é dar naturalidade às nossas traduções.

Por exemplo, em inglês usa-se bastante a passiva, coisa que no português não ocorre com tanta freqüência. “A present was given to me” pode-se traduzir como “Me foi dado um presente”. A equivalência é exata (não vou entrar em discussões lingüísticas) e em um exame de inglês certamente o professor consideraria certa a tradução. Sem problema, se analisamos bem a frase: “quem disse isso?” “quando lhe dão algo você se expressa assim?”. É certo que pode dar ao caso, mas soa muitíssimo mais natural “Deram-me um presente”. Ainda que precise levar sempre em conta o contexto da frase com já foi dito anteriormente, soaria ainda melhor “Deram-me um presente”.

Outra coisa que gostaria de falar é da repetição de informação. Vejamos um exemplo. “Welcome to Aeido. Aeido is a big city!”. O lógico seria traduzir esta frase para “Bem-vindo à Aiedo. Aiedo é uma grande cidade!” Transmite o mesmo sentido que a frase em inglês, mas acredito que “Bem-vindo à grande cidade de Aiedo!” soa muito melhor. Talvez não expresse exatamente igual, mas nem toda frase deve ter uma equivalência exata nem seguir a ordem sintática do texto original. Observemos esse outro exemplo: “Chaz: It's Great! -Alys: I don't think so...”. Uma boa tradução seria: “Chaz: É genial! -Alys: Eu acredito que não...”, mas sinceramente creio que “Chaz: É genial! -Alys: Pois eu acredito que não...” soaria ainda melhor porque é muito mais natural. Não há com o que se preocupar desde que haja exatamente a mesma informação em um texto traduzido, pois às vezes tirar ou acrescentar elementos ao texto original favorece a tradução. Em qualquer caso, toda decisão deve estar justificada, já que somos tradutores e como tais não temos o direito de fazer o que nos der na telha com o texto.

Quanto à tradução de nomes próprios, não há do que ter medo ao trocá-los ou adaptá-los sempre e quando sua troca é justificada. Esta é uma questão um tanto delicada, já que os mais puristas se posicionam contra. Bem, eu proponho o seguinte: o que soa melhor, Cecil ou Cecílio? Anna ou Ana? Eu me calo com Cecil e Ana respectivamente, se bem que é certo que já tenho um contexto criado em mente (mais de um jogador reconhecerá tais nomes). Qualquer que seja a decisão, as explicações estariam contidas no arquivo "leia-me" que sempre acompanha as traduções. Vale lembrar que ver um nome estrangeiro provoca discussões em quem está jogando a tradução, mas também pode dar um ar exótico a ela. Da mesma maneira, adaptar todo nome que aparece em um texto pode implicar numa falta de credibilidade com respeito ao texto original e diminuir a beleza de sua respectiva cultura (já que o idioma esta extremamente unido à cultura). Não devemos esquecer que a atenção dirigida a uma tradução influi muito na mesma.

1.4. A necessidade de adaptações

A tradução audiovisual, e em geral a de jogos de video-games neste caso, caracteriza-se por impor ao tradutor uma série de limitações. A princípio, não se pode adicionar mais texto do que há originalmente em uma conversação. Esta construção poderá ser burlada mediante o uso de diferentes técnicas que serão vistas mais adiante neste manual. Apesar de que, em teoria sempre se poderá ampliar o espaço original, levar tal afirmação à prática pode resultar numa tarefa nada simples que pode desesperar até a pessoa mais paciente dada sua complexidade. Especial importância é cobrada na hora de traduzir os menus que geralmente aparecem em muitos jogos(especialmente RPGs) e cujos impedimentos baseiam-se na limitação de caracteres impostos por uma janela imóvel. Uma solução para tal problema é empregar as abreviações, se bem que não possuem um grande prestígio por diminuir o impacto visual. Ainda assim, até as traduções oficiais já vêm marcadas por tal visão.

Quando traduzimos e vemos que esta limitação de espaço nos obriga a aderir ao espaço marcado, devemos tratar de reexpressar o mesmo conteúdo da mensagem traduzida, mas de uma forma diferente. E então, como já disse anteriormente, considere que um exercício de tradução requer de vez em quando uma reexpressão de sentido. Por exemplo, se não podemos colocar “Não funciona” por exceder um caractere, uma possível solução seria reexpressar o mesmo sentido em “Não serve”. Se bem que pode parecer que “servir” é de uma qualidade estética menor que “funcionar”, mas cumpre perfeitamente sua obrigação e nenhuma crítica deverá cair ao tradutor por optar por esse sinônimo. Casos em que se devam prescindir informação, tratará-se de manter a essência da mensagem. Por exemplo, “Mas o que disse?” passaria para “Que disse?” se assim for preciso. Em efeito, perde-se a raiz, mas continua com o mesmo sentido. Por desgraça, há muitas ocasiões em que não há alternativa que não a eliminação de informação, ainda que por sorte muitas mensagens disponham de informações supérfluas que não acrescentam nada de novo à conversação.

1.4.1. Conselhos ortográficos

Eis aqui uma série de conselhos a levar em conta na hora de traduzir um jogo:

- É melhor colocar duas frases curtas em duas linhas do que uma em uma e meia:
  - ✗ Cecil, vieram te ver ontem. Não tinham boa cara.
  - Cecil, vieram te ver ontem.
Não tinham boa cara.

- É inadmissível usar abreviações, por exemplo o vc no lugar de você:
  - ✗ Vc poderia me ajudar? Você poderia me ajudar?

- Sempre que se pode e não seja uma cifra grande e complicada deve-se escrever um número por extenso.
  - ✗ Deram-me 8 anéis. Deram-me oito anéis.
  - ✗ Faz 1000 anos. Faz mil anos.
  - ✗ Custa 27 Moedas. Custa vinte e sete moedas..

- As reticências não devem ser utilizadas para indicar que há mais texto para ser mostrado na próxima tela. Em tal caso pode ser conveniente adaptar a mensagem para que caiba em um só tela, se possível:
  - ✗ Cecil me disse que viria logo para irmos ao bosque... ... com os outros.
    Cecil me disse que viria logo para irmos ao bosque com os outros.

1.5. Os jogos de palavras.

Um dos grandes desafios que se apresentam aos tradutores são os jogos de palavras. Para esta seção baseei-me em informações de um projeto universitário. Tentarei ser o mais breve possível, pois há muito para se falar sobre o assunto.

Adrián Fuentes Luque defendeu uma tese que tratava sobre a recepção do humor em um estúdio que fizeram o filme "Sopa de Ganso" dos celebres Irmãos Marx. O filme não era mais do que um desafio de titãs para os tradutores. Um dos fragmentos mais interessantes do filme é o seguinte:

- FIREFLY (Groucho Marx): *I suggest that we give him **ten** years in **Leavenworth**, or **eleven** years in **Twelveworth***.
- CHICOLINI (Chico Marx): *I tell you what I'll do. I'll take **five and ten** in **Woolworth***.

Aqui o humor se baseia na decomposição do nome próprio *Leavenworth* (uma prisão federal dos Estados Unidos) em duas partes. *Eleven* [onze] e *worth* [valor], pelo que se estabelece numa relação fonética com os elementos *Twelve* [doze] e *worth de Twelveworth* (uma palavra inventada por Groucho) e com *Woolworth* (uma cadeia de tendas americanas pela sua campanha de *five and ten*). Além disso, um jogo de palavras com a seqüência de números de *ten years* in *Leavenworth, or* ***eleven*** *years in* ***Twelve****worth* e em ***five*** *and* ***ten*** *in Woolworth*. A solução proposta na versão dublada em espanhol do filme é a seguinte:

- FIREFLY (Groucho Marx): Sugiro condená-lo a **dez** anos de **cárcere** ou **onze** anos de **prisão**.
- CHICOLINI (Chico Marx): Sabe o que farei? Tomarei umas férias no campo.

De certo modo, a versão original tem pouco a ver com a dublada, as estratégias de traduções empregadas são notáveis. Em primeiro lugar, deve-se destacar que sem alternativas, toda referência cultual desapareceu. Contudo, nesta tradução optou-se pelo absurdo como meio de humor. Não é preciso mais que ler a frase de Chicolini para soltar uma gargalhada, tal como disseram os participantes do estúdio. E disso se trata: o diálogo original tinha como propósito divertir o expectador, algo que pode ser comprovada na versão em espanhol. De fato, a versão legendada está marcada por uma literatura que nenhum expectador achou graça na cena, sendo que muitos acharam-na estranha.

1.6. Dicionários e outras fontes de informação

É mais provável que não seja uma pessoa bilíngüe e não conheça com perfeição o idioma estrangeiro que irá traduzir. Por isso, não há de se "envergonhar" com o uso de dicionários. Todo tradutor usa, mas sempre com um cuidado especial. Não é preciso abusar, e menos ainda confiar totalmente neles. Por exemplo, sei que existe muitas traduções de jogos em que se traduz "pendant" por "pendente". A mim também já aconteceu; e conto adiante da minha história. Ainda que sabendo que "pendant" era "pendurado", coloquei minha confiança no dicionário e optei por usar "pendente". Sim, efetivamente, o dicionário colocava isso. Há de se usar o senso comum e ver se o que diz o dicionário pode ser considerado "verdadeiro". Muitas vezes é melhor analisar o contexto (espero que com isso fique totalmente clara sua função) e ver que palavra ou expressão é a que melhor encaixe e usar o dicionário para assegurar-se e constatar a possibilidade.

O dicionário bilíngüe (aquele em que se busca uma palavra em um idioma e vem seu significado em outro) não é sempre a única solução, às vezes é recomendável recorrer ao dicionário monolíngüe (a explicação vem dada no mesmo idioma, por exemplo, qualquer dicionário de português que usamos quando não entendemos alguma palavra complexa). Caso se traduza desse idioma estrangeiro não deve haver problema para compreender a explicação. A propósito, hoje em dia muitos dicionários estão disponíveis em CD-ROM para usá-los nos computadores, coisa que recomendo encarecidamente, já que assim é muito mais fácil e cômodo de buscar palavras e até permite várias opções de busca. Não só se pode usar dicionários; pergunte a alguém cuja língua mãe é a que nós ainda tratamos de aprender, pode ser uma boa solução, igual a internet. Contarei outra de minhas experiências.

Um vez, encontrei numa série de anime legendada em inglês que eu traduzia para o espanhol uma expressão que não entendi: “Serves you right!”. Tal expressão era falada em um momento delicado e não podia inventar a expressão. Como o dicionário não me ajudava, fui ao canal de IRC onde conversava um grupo de tradutores ingleses e, sem preguiça ou timidez, os perguntei o que significava tal expressão. Em dez segundos me responderam, e pela explicação que me deram, cheguei a conclusão de que deveria traduzi-la para "Você merece!".

1.6.1. Dicionários recomendados

É evidente que não tenho em minhas mãos todos os dicionários que existem, ainda assim farei uma recomendação. Como dicionário bilíngüe inglês-espanhol e espanhol-inglês, eu uso o Gran Dicionário Oxford (Terceira Edição). Vem com um CD-ROM (realmente facilita a árdua tarefa de procurar palavras na versão impressa). Eu gosto, é muito completo, ainda que o dicionário de Collins não pareça ser uma má opção. Como dicionário monolíngüe de inglês uso a versão em CD-ROM do Merriam-Webster's 11th Collegiate Dictionary. A verdade é que é muito completo e sempre é encontrado tudo o que se procura. Se traduz do francês, eu uso como dicionário bilíngüe a versão impressa (não encontrei nenhum decente para computador) do Gran Diccionario Larousse e como monolíngüe a versão em CD-ROM do Le Petit Robert, Se alguém traduz do alemão, usar dicionários da editora Langenscheidt parece ser a melhor solução.

1.7. Erros mais comuns nas traduções

Fiz aqui uma pequena lista de erros que tenho visto em algumas traduções de ROMs:

- *Actually*: Não, não é "atualmente". Eu sempre opto traduzi-lo por "de fato", "a verdade é que..." ou expressões parecidas.
- *Bastard*: Sim, pode se referir a um filho bastardo, mas se emprega também como insulto. Supondo que "corno" é muito forte para colocar em um jogo, cabe ao tradutor decidir colocá-la ou não.
- *A ordem do adjetivo*: Em inglês se antepõe o adjetivos do substantivo. "He's a strong man" deve ser traduzido para "é um homem forte" e não por "é um forte homem". Em muitos casos pode-se traduzir dessa maneira, mas deve ter cuidado.
- *Eventually*: Não se deve traduzir por "eventualmente", e sim por "finalmente", "ao final" ou expressões similares.
- *Now*: Igual ao anterior, é um dificuldade. Eu traduzo por "Bem".
- *Pendant*: Como foi dito mais acima, não deve traduzí-lo para "pendante" e sim por "pendurar"
- *Surgir efeito*: O correto é "Surtir efeito".
- *Virtually*: Não tem nada a ver com virtual. Se deve traduzir para "praticamente" ou similares.
- *What on earth?*: "O que acontece na terra?" não, deve traduzí-la para "Que demônios?" ou expressões parecidas.
- *Why*: Nem sempre é "por que", já que se pode empregar como interjeição de surpresa, é dizer, que se deve traduzir para "anda", "vá"...
- *Ya*: Contração de "you", não tem nada a ver com o "ya" espanhol.
- *Yo*: É um tipo de saudação muito coloquial (é como dizer "como vai" entre amigos).

CAPÍTULO 2:
INTRODUÇÃO AO ROMHACKING

Neste capítulo serão explicados os passos básicos necessários para começar a editar o texto de um jogo qualquer. Em outras palavras, verão o processo utilizado para procurar texto em um arquivo binário de um jogo (geralmente trata-se de uma ROM).

2.1. Programas necessários e outros requerimentos

- *TaBular*: Programa que simplifica o trabalho de fazer uma tabela manualmente. Mais adiante veremos o que é um tabela.
- *Editor de texto* (Bloco de Notas do Windows por exemplo): Servirá para modificar a tabela de uma forma rápida.
- *SearchR X*: é capaz de procurar texto em arquivos muito grandes (como ISOs de PSX) sem necessidade de carregá-lo na memória RAM, algo que pode deixar lento o processo de abrir o arquivo. Além disso, permite fazer uma busca de 16 bits.
- *O arquivo que se deseja traduzir*. Tomarei como exemplo a ROM de Secret of Mana, já que é uma ROM que não oferece problemas na hora de procurar o texto.

2.2. Conceitos Básicos: código hexadecimal e tabelas

Não quero me alongar mais que o necessário, gostaria de esclarecer da melhor forma possível estes dois termos já que qualquer tradução que façamos, faremos uso freqüentes deles.

2.2.1. Código hexadecimal

Hexadecimal faz referência a um sistema de numeração que, tal como indica seu nome, toma um inteiro de base 16. Então, o número 10 em decimal equivale a 0A em hexadecimal. Cada número em hexadecimal é representado por dois dígitos que podem ir de 00 à FF (255 em decimal). Abaixo podemos ver uma série de números decimais com seus correspondentes valor em hexadecimal.

Sistema decimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18...
Sistema hexadecimal: 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10, 11, 12...

A pesar de que talvez o correto seja dizer “valor hexadecimal”, tende-se a ampliar mais ao termo “código hexadecimal” para falar sobre números hexadecimais ou simplesmente “código hex”.

Além do sistema decimal e hexadecimal existem mais dois: o sistema binário e o octal. Por ser provável que trabalhemos com o sistema binário, procederei com sua explicação. Neste sistema cada número é composto de 8 bits, quero dizer, oito dígitos, que podem ser 0 ou 1. Por exemplo, 00000011 em binário corresponde ao número 3 em decimal. Caso seja preciso fazer conversões de um sistema para outro, existem muitas calculadoras que nos podem ser de grande ajuda, como a que o Windows traz (vale lembrar que deve ativar o modo científico).

2.2.2. Tabelas

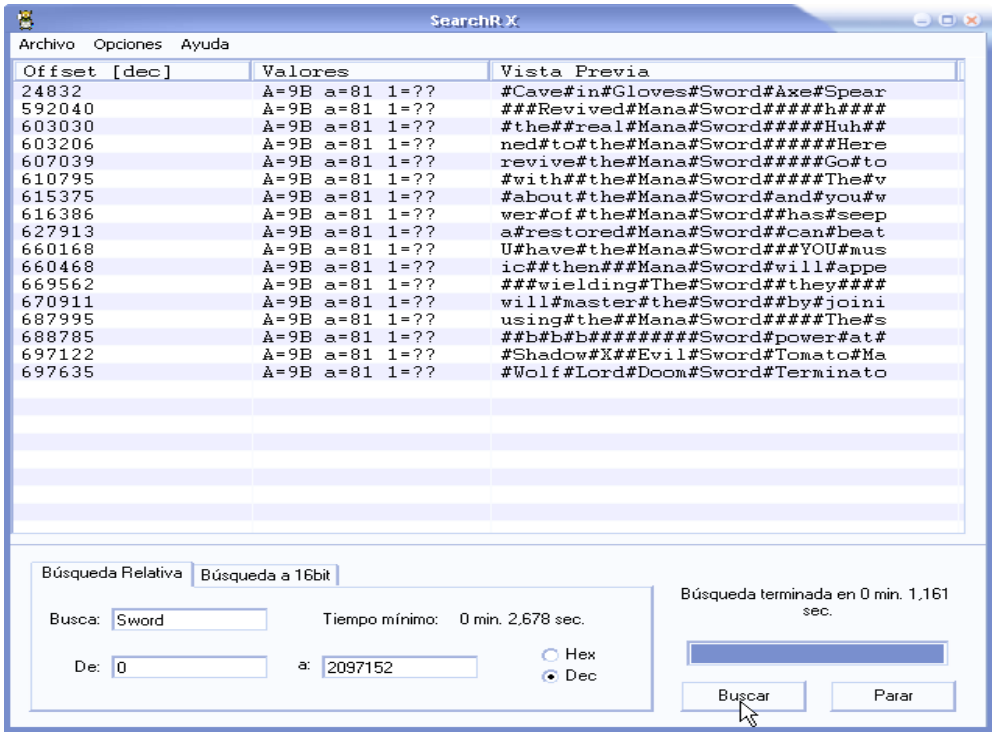
Uma tabela é um arquivo de texto com extensão .tbl no qual se especifica a equivalência dos caracteres que um jogo usa (a, b, c...) em hexadecimal. Também se utiliza tal termo para referir-se ao conjunto de equivalências que estão em uma tabela sem fazer referência ao arquivo com extensão .tbl. As tabelas são usadas por programas como Thingy ou Translhextion com o objetivo de mostrar neles os caracteres latinos (no caso de traduzir do inglês) que previamente especificaremos. Desta maneira será possível ver e editar o texto do jogo em questão mais comodamente.

O valor de cada igualdade é a seguinte: (Valor hexadecimal)=(caractere). Por exemplo, algumas igualdades da tabela do Secret of Mana são:

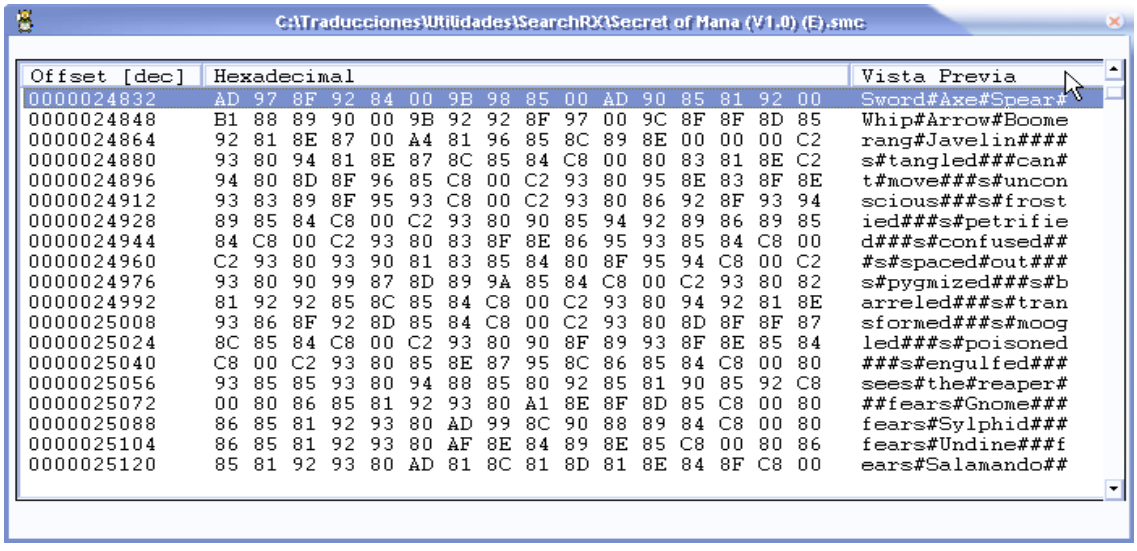
82=a
9A=z
9C=B
...

2.3. Procura por textos em um arquivo binário

Para isso a primeira coisa que temos que fazer é executar o SearchR X e carregar a ROM do Secret of Mana. Uma vez carregada, nos situamos na caixinha de texto e escrevemos uma palavra que apareça no jogo, como por exemplo “Sword”. Note que essa palavra possui maiúsculas e minúsculas. Clicamos no botão *Buscar* e quando terminar olhe para a área de baixo onde tem os *Valores*. Veremos que existem várias entradas dentre elas aparecem A=9B e a=81 como mostra a seguinte imagem:



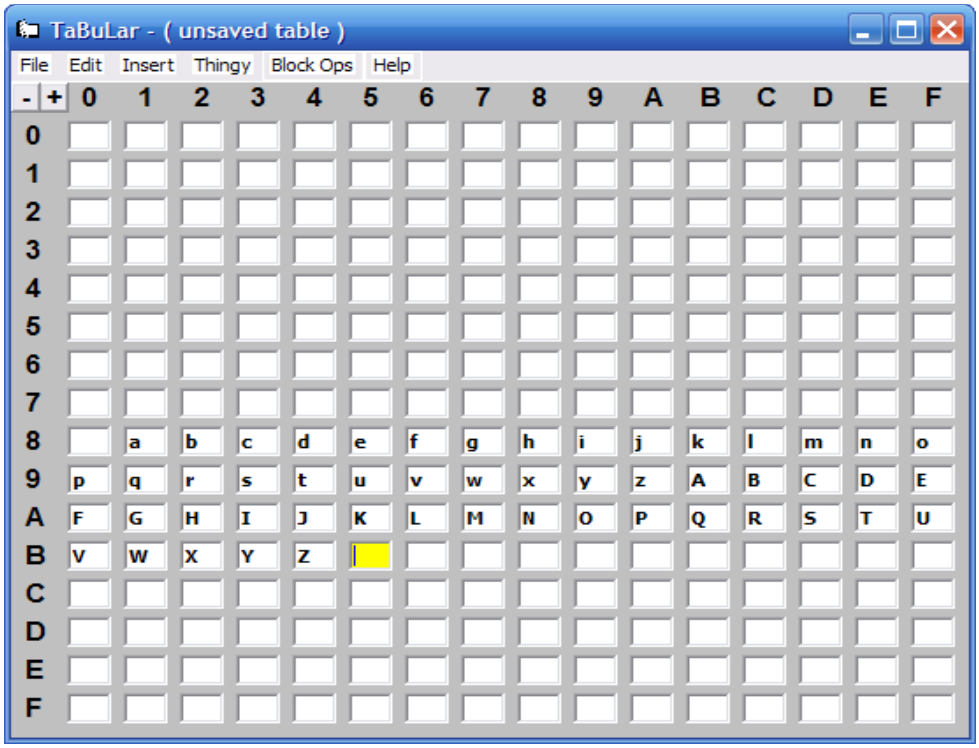
O que aparece em *Offsets* é irrelevante neste caso (um offset indica a posição do texto dentro do arquivo binário). O que surge em *Vista Prévia* nos permite ver o texto usando a equivalência dos valores como se de uma tabela se trata. Ao que parece, tais equivalências são as que são usadas no jogo, já que todos os textos em que "Sword" aparece usam os mesmo valores e estão rodeados por outras palavras em inglês. Se queremos nos assegurar disso, podemos dar um duplo clique na tela do programa onde mostra tudo e aplicar *ASCII* para que mostre o texto com os valores da possível tabela:



Agora que sabemos a tabela do jogo, basta lembrar o valores de “A” e “a” para fazer a tabela. Porém, caso não tenha encontrado os textos no caso de ter tentado com outro arquivo que não seja o Secret of Mana. O mais provável é que o texto esteja comprimido; no capítulo 5 serão dados mais detalhes sobre a procura de textos comprimidos.

2.4. Fazer uma tabela

Para isso devemos usar o TaBuLar. O que temos que fazer é colocarmos o cursor na casa do valor hexadecimal do caractere “A”, que no caso do Secret of Mana é 9B. Os números e letras da esquerda são para o primeiro dígito do código hexadecimal(9) e os de cima do segundo (B). No menu “*Insert*” colocamos em “*Capital English Alphabet*”. Agora colocamos o valor que corresponde ao “a” e fazemos o mesmo, só que dessa vez colocamos “*Lowercase English Alphabet*”. Salvamos a tabela no formato do Thingy e pronto. O resultado deverá ser como o da imagem a seguir:



Desta maneira temos uma tabela com os valores hexadecimais do alfabeto latino, ainda que falte alguns outros caracteres, como os sinais de pontuação. No próximo capítulo veremos de qual maneira de acha o valor hexadecimal de tais caracteres.

2.5. Para saber mais

- *The Definitive Guide to ROM Hacking Tables* (por InVerse): Excelente documento no qual se explica detalhadamente todos os passos para fazer uma tabela. Além de explicar com fazer as tabelas de jogos japoneses.

CAPÍTULO 3:
EDIÇÃO DE TEXTO

Neste capítulo veremos como editar o texto de uma ROM a partir da base que espero que tenha se consolidado depois de ler o capítulo anterior. Editar o texto será uma tarefa fundamental durante o processo de tradução, veremos ainda mais de um modo de traduzir o texto de uma ROM.

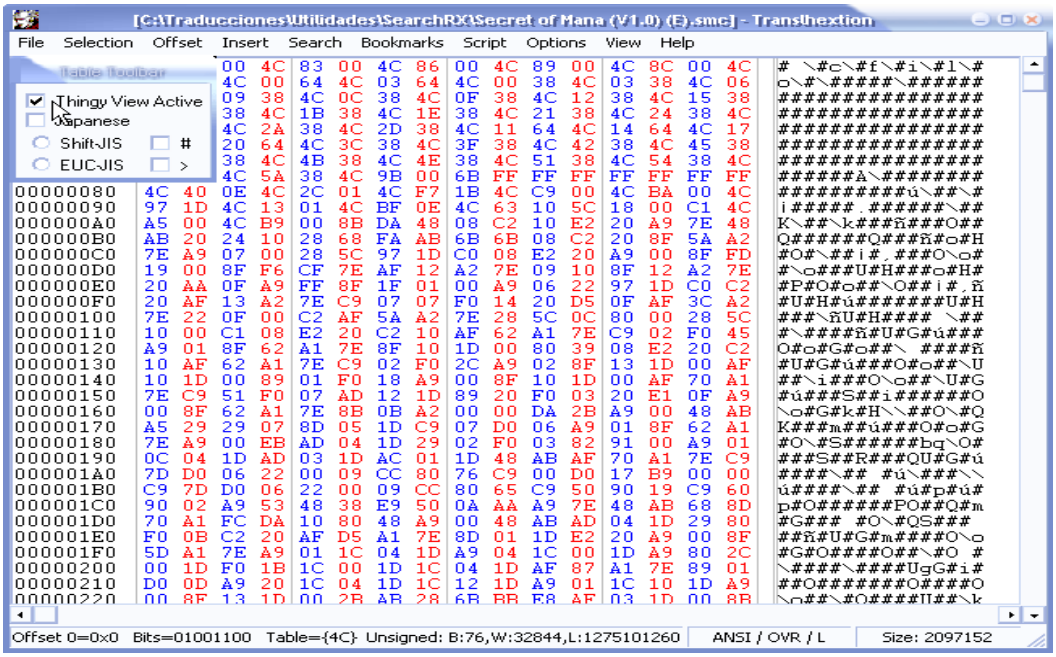
3.1. Programas necessários e outros requerimentos

- *Translhextion*: Além de ser um potente editor hexadecimal, considero-o como o melhor programa para a extração e inserção de scripts (que veremos mais adiante) apesar de algumas carências.
- *ThingyV*: É a versão modificada por Vegetal Gibber da Vegetal Translations que acrescentou algumas características novas ao clássico Thingy, tais como suporte para arquivos grandes (como ISOs de PSX) e uma opção no caso de precisar usar duas tabelas.
- *O arquivo que deseja traduzir*. Tomarei como exemplo a ROM do Secret of Mana..

NOTA: Verifique que a ROM não tenha o atributo Somente Leitura, caso contrário, ao salvar as modificações acontecerá um erro. Para verificar , dê um clique com o botão direito sobre o arquivo da ROM e vá em Propriedades. Assegure-se de que a opção Somente Leitura NÃO esteja marcada, tal como é mostrado na imagem:

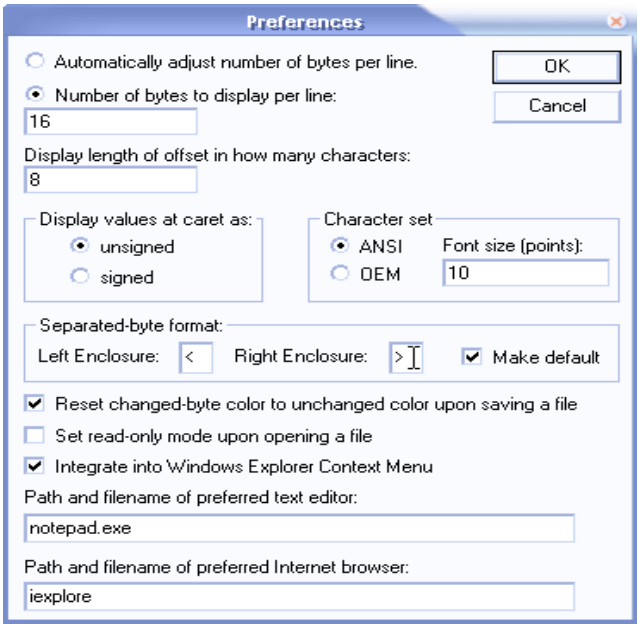
3.2. Uso do Translhextion

No capítulo anterior fizemos uma tabela que incluía todo o alfabeto latino, mas que não estava completa faltando caracteres como os sinais de pontuação. É necessário completá-la antes de começar a traduzir, para isso devemos fazer uso do programa Translhextion, e já que vamos usá-lo constantemente, explicarei todas as suas funções ainda que pareça exaustivo.



- No menu *File* é onde abriremos os arquivos para modificações, as demais opções não necessitam de explicação.
- No menu *Selection* Selection podemos encontrar os típicos comandos para copiar, cortar e colar o texto.
- No menu *Offset* Offset está o super útil *Jump* o que nos servirá para ir a posição ou endereço (offset) que queiramos. Se colocarmos um endereço em hexadecimal será necessário colocar um x antes do endereço (por exemplo, x10EE). Note que não é preciso colocar os zeros da esquerda.
- No menu *Insert* não tem muitas opções úteis para dizer a verdade.
- No menu *Search* introduziremos o texto para fazer uma busca utilizando a tabela. É muito sensível como veremos. Também pode-se procurar por texto em ASCII (texto que não precisa de tabela para mostrar corretamente os caracteres). As demais funções são interessantes, mas para isso já temos o SearchR X.
- No menu *Bookmark* podemos inserir na tabela, *marcadores de offset*, quer dizer, uma espécie de acesso direto para uma posição determinada do arquivo binário. É muito útil colocar marcadores em diversas partes do texto que pode ter no jogo, como os menus, créditos ou o texto principal. Translhextion suporta até 9 marcadores.
- No menu *Script* podemos encontrar tudo o que é relacionado aos scripts. Veremos mais detalhes sobre scripts mais tarde.
- No menu *Options* encontramos as *preferências* do programa. Recomendo trocar os **{ }** por < >

como pode ser visto na imagem seguinte porque os scripts (logo veremos o que eles são) extraídos com outros programas usam <>, o facilitará a compatibilidade. Além disso, se desejar que ao clicar em um arquivo com o botão direito sai no menu contextual a opção *Abrir com Translhextion*, ativada a opção *Integrate into Windows Explorer Context Menu*

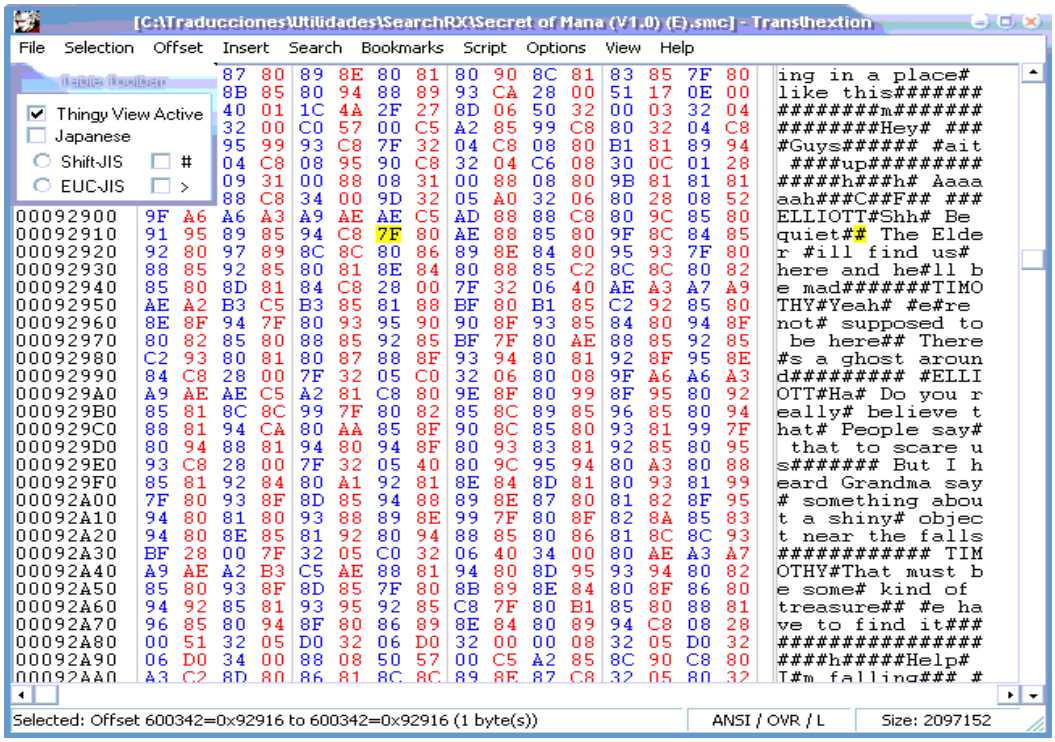


3.3. Completando a tabela

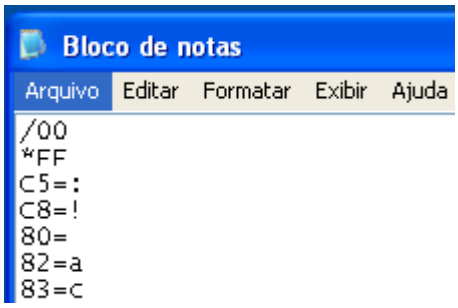
Para isso devemos saber o texto exato de algum diálogo, então jogaremos um pouco o Secret of Mana para ver e trabalhar com um dos primeiros textos. É escolhido o que se mostra na imagem abaixo:

Nossa tarefa será procurar na ROM a frase *Be quiet*. Para isso, executamos o Translhextion e carregamos tanto a ROM como a tabela que previamente criamos. Vamos ao menu *Search* e escolhemos a *Find Using Table*. Procuramos *Be quiet* e em poucos segundos estaremos diante do texto. Todos os # que são vistos ao redor do texto são os códigos hexadecimais que não constam na tabela. Nosso objetivo é procurar os valores que nos faltam, não precisa descobrir pra que servem TODOS os #.

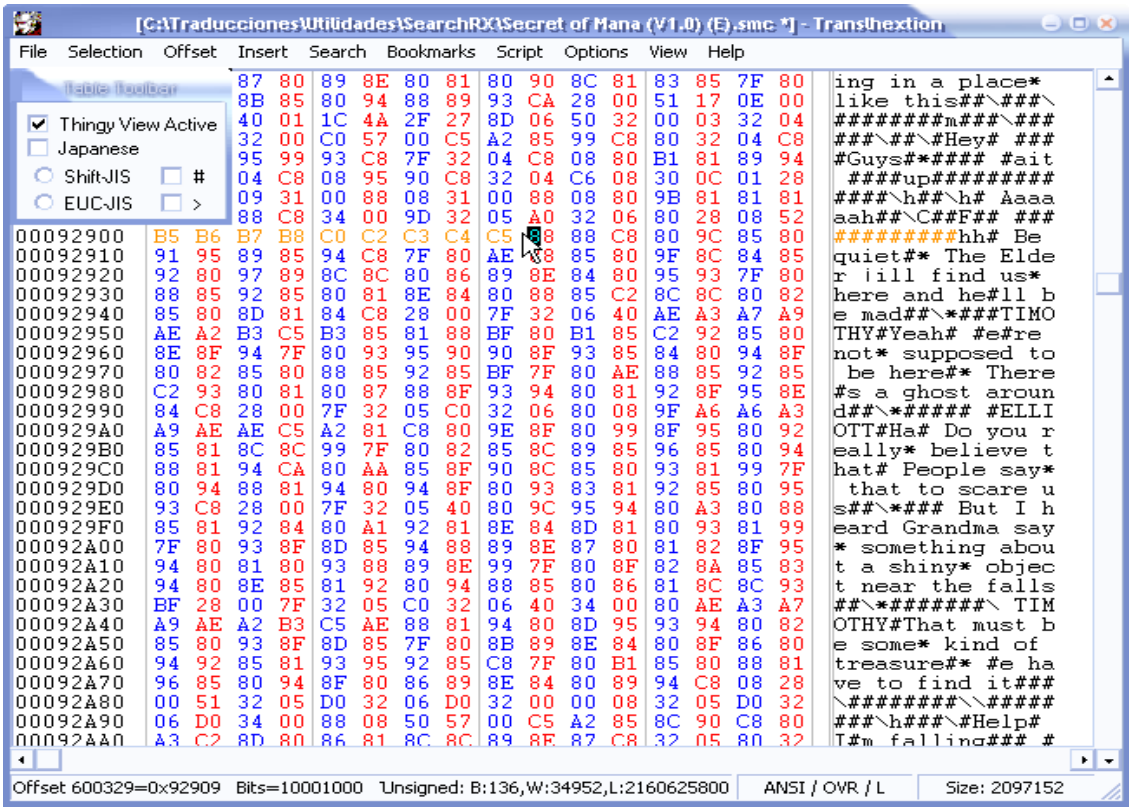
Bem, coloque o cursor no # que esta logo depois de *ELLIOTT* e olhamos seu código hexadecimal a esquerda. Nos baseando no texto que vimos antes, deduzimos que C5 equivale a (:) e que C8 a (!). Também há outros # interessantes no texto, como o que está ao lado do primeiro ponto de exclamação. Este código, cujo valor é 7F, indica quebra de linha. Também é importante o código que está atrás do último ponto de exclamação. já que aparentemente representa o código do fim da mensagem. Porém, temos que ressaltar que Secret of Mana possui vário códigos entre diálogos e diálogos, e depois de analisar outros veremos que o verdadeiro código de fim de mensagem é o 00. De fato, muitos jogos usam este valor como código de fim de mensagem. Por outro lado, parece evidente que o código 80 representa um espaço. Aqui pode ver uma imagem de área da ROM de Secret of Mana.



Bem, o que temos que fazer agora é abrir a tabela como o Bloco de notas do Windows (ou qualquer outro editor de texto) e acrescentar esses valores, o código de quebra de linha se escreve com o um asterisco (*) e de fim de mensagem com uma barra (/). Desta forma, a tabela deveria ficar assim:



Agora é preciso procurar por outros diálogos para acrescentar vírgulas e outros caracteres, um outro método eficaz de encontrar vário códigos hexadecimais e suas equivalências é ver seu valor logo no jogo. Lembre-se que para editar os códigos hexadecimais que aparecem a esquerda temos que usar o TAB.



Por último, lembre-se que se fazer alterações na tabela deve carregá-la novamente no Translhextion para vê-los.

3.4. Edição do texto

Já estamos capacitados para poder editar o texto de um arquivo. Editar o texto com o Translhextion não tem mistério, se bem que é surpreendente que tem havido muitas pessoas que têm problemas para entender o modo de edição. A única coisa que há de se fazer é situar-se no caractere a partir do qual se deseja traduzir e apertar a tecla <ENTER>, deste modo aparecerá uma janela de edição do Translhextion.

Eu tenho outra forma editar o texto com o Translhextion baseada na maneira em que se faz no Thingy. Vá ao dialogo que queira traduzir, situe-se no último caractere que deseja editar e vá até o primeiro caractere do diálogo ou frase que deseja traduzir selecionando o texto (com a tecla shift).Preste atenção nos bytes selecionados e aperte a tecla <ENTER>. Quando terminar de traduzir volte a teclar <ENTER>, ainda observando os bytes selecionados para não haver problemas. Se sobrar caracteres, simplesmente preencha os bytes restantes com espaços. Lembrando de fazer cópias de segurança dos arquivos que se edita com freqüência é sempre um bom costume. Levando em conta o número máximo de caracteres que pode ter uma linha e quantas linhas pode ter um caixa de texto, já que se os exceder poderá provocar uma série de conseqüências catastróficas.

CAPÍTULO 4:
EDIÇÃO DE GRÁFICOS

É mais que provável - especialmente se, além disso, já tentou traduzir um jogo - que já tenha se perguntado durante a leitura deste manual, o por que de os acentos não aparecerem em canto algum dos jogos. Talvez sequer tenha pensado na possibilidade de que não exista nenhum problema relacionado com isso, algo também bastante óbvio. Bem, o que acontece é que nos jogos em inglês (ou japoneses, a questão é que não estejam em português) não existem tais caracteres, simplesmente porque não são utilizados. Por tanto, se queremos escrever uma letra tão representativa de nosso idioma como o "ã", antes devemos inserí-la na fonte que é utilizada pelo jogo.

Os editores gráficos permitirão realizar essa tarefa que requer trocar alguns gráficos por outros que não utilizemos na ROM, como por exemplo o apostrofo ('). Depois de tudo, toda a fonte da ROM se baseia em um conjunto de gráficos, assim, a ROM se limitará em mostrar os gráficos que estão armazenados em um determinado lugar do arquivo. Durante todo este manual farei referência a cada quadradinho que aparece em um editor gráfico com *tile* já que é esse o nome comumente dado pelos *expert* no assunto (é preciso ter em mente que muitas palavras são plágios tomados desnecessariamente do inglês).

Por outro lado, tenho que dizer que trocar gráficos não é como retocar imagens com um programa como pode ser o Paint do Windows, pois todos os gráficos de um arquivo aparecem "desordenados". A primeira vez que carregar uma ROM em um editor gráfico terá a impressão de que está tudo corrompido quando ver todos os gráficos da maneira que aparecem. Algumas vezes é mais fácil encontrar a fonte do jogo, mas outras vezes não será tão simples porque podem estar comprimidas ou usar um formato de tiles desconhecido, para citar algumas possibilidades.

4.1. Programas necessários e outros requerimentos

- *Tile Layer Pro*: Suporta quase todos os formatos de ROMs e possui um muito útil *clipboard* no qual se pode arrastar os tiles "desordenados" de maneira que a edição de gráficos seja feita mais fácil e comodamente.
- *Tile Molester*: Suporta ainda mais formatos que o Tile Layer Pro e permite "arrumar" gráficos como veremos mais em seu devido momento, embora para mim seja mais fácil editar os gráficos com o Tile Layer Pro. Foi programado em Java, por isso talvez seja necessário instalar as bibliotecas de execução do Java, pois sem elas não será possível executá-lo.

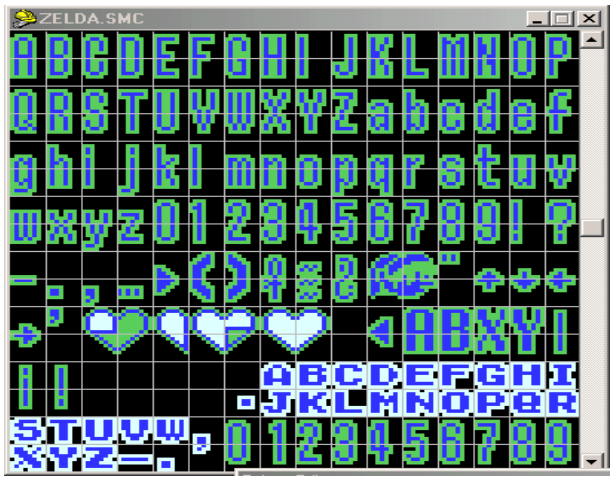
- *O arquivo que se deseja alterar os gráficos*. Tomarei neste caso como exemplo a ROM de Zelda do SNES e a ROM francesa de Final Fantasy Mystic Quest.

4.2. Editando gráficos

4.2.1. Gráficos visíveis

O que faremos primeiro será executar o Tile Layer Pro e abri a ROM de Zelda de SNES. Em seguida devemos clicar em *Format* no menu *View* e selecione o modo gráfico adequado. O lógico será utilizar o modo SNES neste caso, será? Nada mais distante da realidade. Resultado irônico, pois em algumas exceções é preciso escolher um formato que não corresponde com o sistema do console. O mais normal é que um jogo de SNES use o formato do Game Boy para a fonte de uma ROM, inclusive o modo *1bpp*. Por tanto, é recomendado optar por estes dois modos antes que qualquer outro para visualizar a fonte.

Uma vez dito isso, agora é só ir deslocando para baixo no editor. Enquanto faz isso você poderá ver de vez em quando entre muita desordem alguns gráficos do jogo em questão. Continuando encontrará lá pela metade da ROM de Zelda de SNES uma imagem como o abaixo:

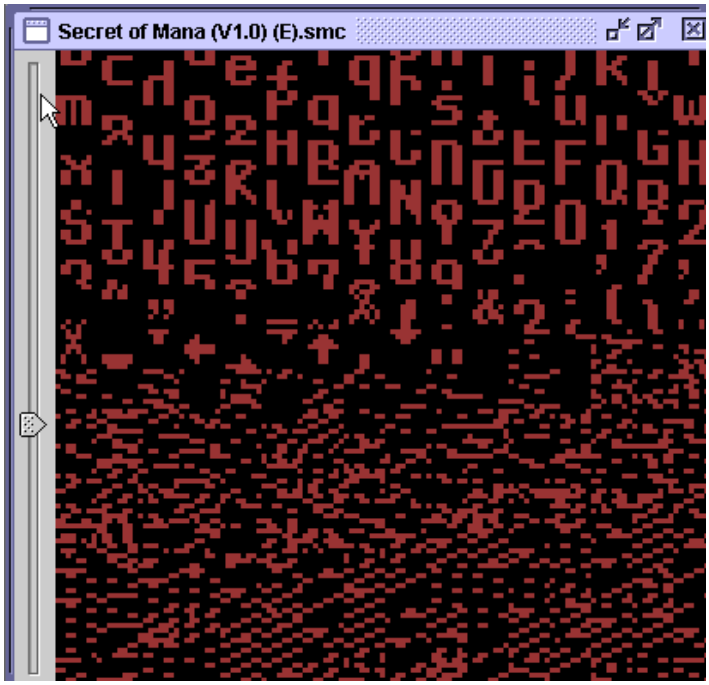


Se quer trocar, por exemplo, o apostrofe(') que está situado depois da flechinha que assinala à direita e antes do primeiro pedaço do coração pelo "ã" basta selecionar o tile em questão e, na janela cujo nome é *Tile Editor* podemos alterá-lo de acordo com nosso gosto. Para desenhar um "ã", a melhor opção é pegar o tile "a" - que é igual ao "ã" exceto pelo acento - arrastando o tile para a posição do apostrofe e assim editá-lo economizando esforços.

Quando aparecer tal caractere na tela enquanto se joga, o que aparecerá agora será o que nós desenhamos. Se editar um tile cujo valor não tem na tabela, simplesmente some o tile com o último tile que tenha na tabela. Para que fique mais claro, vamos ao seguinte exemplo. O símbolo que fecha parênteses “)” é o último caractere e este possui o valor hexadecimal de 46. Se editar esse estranho símbolo que aparece justamente antes do rosto do Link, o “e” comercial (&), contamos desde o 46 um a um até chegar a esse tile e obteremos como resultado o 49.

4.2.2. Gráficos que utilizam um tamanho diferente do padrão

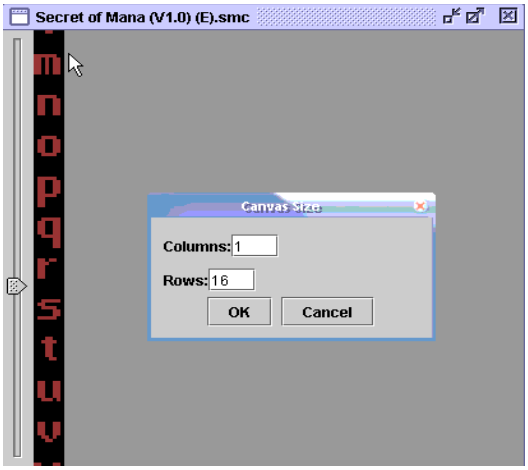
É muito provável que não tenha encontrado a fonte de uma maneira clara, mas tenha certeza que se trata dela. Você a verá tão desordenada que será praticamente impossível editá-la com facilidade. Isto acontece, por exemplo, se tentarmos editar a fonte do Sercret of Mana com o Tile Layer Pro como se pode ver nesta imagem (que foi tirada como o Tile Molester):



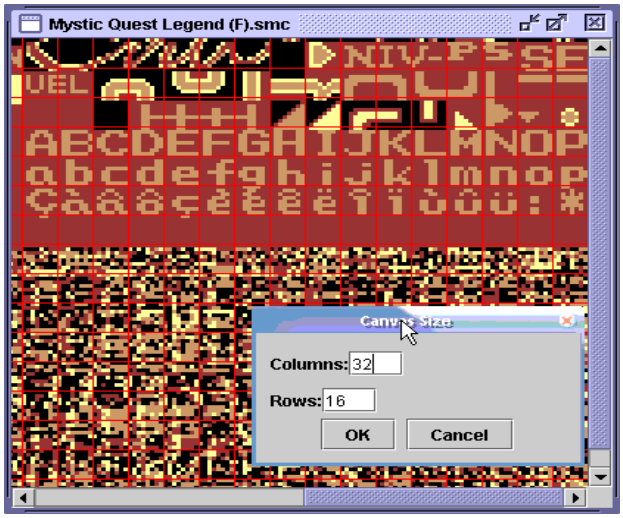
Para poder editar fontes que tenham tal inconveniente deveremos usar o Tile Molester. Supondo que já tenha instalado as bibliotecas de execução do Java, o executaremos. Para isso, abra uma janela do MS-DOS (no Windows XP esta no *Iniciar* > *Todos os programas* > *Acessórios* > *Prompt de Comando*) vá até a pasta (lembre de digitar *cd* <*pasta*> para andar pelas pastas) onde esta o Tile Molester e digite *Java -jar tm.jar* como se vê na imagem:

Claro que, também se pode criar um arquivo .bat com tais comando para agilizar sua execução por precisar de apenas um duplo clique sobre ele. Simplesmente bastará escrevê-los em um arquivo de texto e salvá-lo com extensão .bat.

Não há com o que se preocupar caso demore para carregar, já que isso é algo normal. Uma vez carregado, abrimos a ROM de Secret of Mana e vamos a posição onde vimos a fonte com o Tile Layer Pro. Embora a fonte de Secret of Mana possa ser vista de leve no modo do Game Boy (codec *2bpp* no Tile Molester, ela é melhor vista em *1bpp*. A grande variedade de codecs do Tile Molester pode ser escolhida no menu *Codec* do menu *View*. É conveniente ativar o Tile *Grid*) do menu *View* para ver um grade que separa os tiles, embora neste caso não nos será de grande utilidade. Agora só nos cabe usar as teclas direitas e esquerdas do teclado mantendo apertada a tecla Shift para trocar as colunas. Também podemos trocá-las em *Image* > *Canvas Size*.



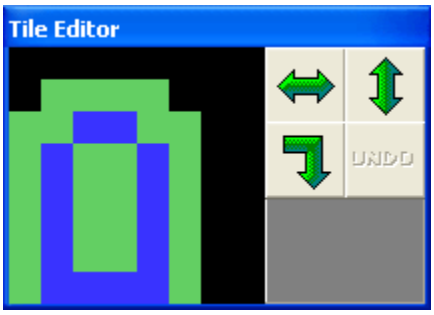
Vejamos outro exemplo. Desta vez carregaremos a ROM francesa do Final Fantasy Mystic Quest de SNES. Se usarmos o Tile Molester e avançarmos pela ROM nós encontraremos a fonte, se bem que neste caso é certo que se visualizará melhor usando o codec *2bpp*. Use o *Tile Grid* que será muito útil nesse caso. A fonte se encontra muito obscura, para tentarmos vê-la melhor selecione o *Mode 2 Dimension* no menu *View*. A fonte ficará muito mais legível mas ainda assim não poderá ser edita com facilidade. Para poder vê-la como vemos a fonte de Secret of Mana, o que devemos fazer é trocar as colunas do *Canvas Size* para 32 como se vê na imagem:



4.3. Alterar a largura de uma fonte de largura variável(VWF)

Para explicar esta separação tomaremos de novo como exemplo o Zelda de SNES. Este jogo utiliza o que se denomina *VWF* (*Variable Width Font,* quero dizer, *Fonte de largura Variável*), o que quero dizer é que no jogo não se apresentará o sinal de exclamação (¡) como pode ser vista no editor, sinal que mostrará os somente os pixels que ocupa. Para isso, a ROM contém uma área ou tabela como normalmente é denominado na qual armazenam-se os valores de largura de cada caractere. Deste modo, se substituir o 'W' pelo sinal (¡), este será mostrado na tela junto com vários pixels vazios ao redor, já que utiliza a largura do 'W', cirando um feito nada vistoso:

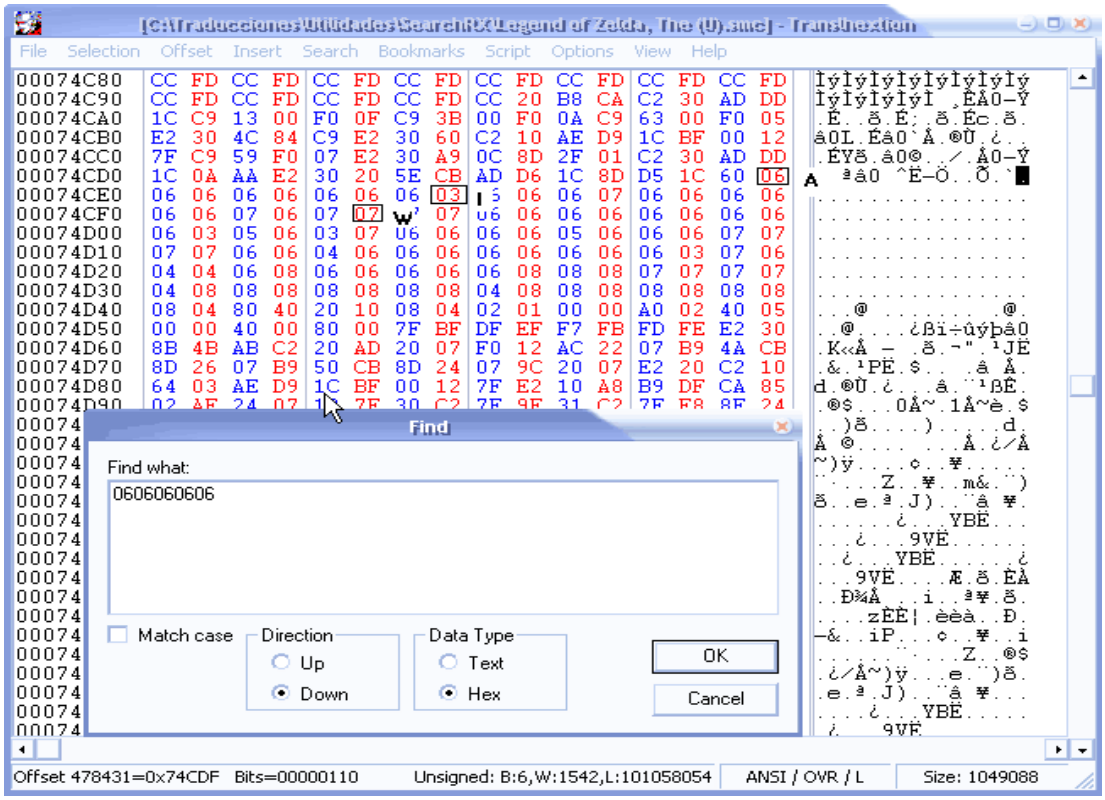
Vejamos como podemos resolver esse problema. Abrimos o Tile Layer Pro, carregamos a ROM do Zelda nos dirigimos ao local onde está a fonte. Prestamos atenção nos pixels que cada tile ocupa. Vamos pegar o tile da letra “A”. Na janela *Tile Editor* podemos ver claramente de quantos pixels é formado este tile:



Podemos observar se contarmos os pixels da esquerda para a direita que são 6 pixels os que formam o tile. Muitas vezes o último pixel de um caractere é um espaço, ainda que em Zelda não seja esse o caso. Do contrário, em vez de 6 pixels falaríamos de 7. Praticamente todos os demais caracteres ocupam os mesmos pixels, salvo a letra 'I' ou o 'j' para citar exceções.

Agora executamos o Translhextion e carregamos a ROM de Zelda. O que vamos fazer é procurar os códigos que representam o número de pixels que mostrados por tile. Como já foi dito que cada caractere é composto em geral de 6 pixels, converteremos tal cifra para hexadecimal, quero dizer, 06. E damos um *Find* no menu *Search* (lembre-se de ativar a procura por hexadecimais) e buscamos 06 06 06 06. A razão disso é que na suposta tabela de valores de largura de cada caractere “abcd” conterá esses valores tendo em conta da largura dos tais caracteres.

É muito provável que se encontrem nesta seqüência, ainda se não é o caso deveríamos provar a busca por 07 07 07 07 (como foi dito anteriormente, alguns caracteres possuem um pixel a mais que serve como espaço). Lembre-se que deve haver muitos códigos parecidos ao redor, já que nem sempre vamos encontrar o que realmente procuramos de primeira. Uma vez encontrado o que procurávamos, veremos que há alguns 03 (3 são os pixels do 'i') entre os 06. Se contarmos quantos caracteres existem entre o 'I' e o 'A', ambos inclusive, veremos que existem 9. Bem, se contamos 9 a partir do primeiro 06 nos daremos conta de que, efetivamente, correspondem ao alfabeto. Agora fazemos o mesmo, mas entre o sinal de exclamação (¡) e o 'A', que é o que nos interessa. Se formos substituir o 'W' (que possui 7 pixels) por esse sinal, o que devemos fazer é procurar a largura do 'W', que esta a 22 caracteres do 'A'. O sinal (¡) possui 3 pixels, assim que trocarmos o valor original do 'W', que é 07, por 03. Tudo isso pode ser visto mais claramente na seguinte imagem:



Uma vez terminado, salvamos as alterações. Se tudo for alterado corretamente, agora aparecerá o tile como deveria, como podemos ver nesta imagem:

4.4. Para saber mais

- *Console Graphics Document* (por Klarth): Vem explicando o formato de cada console. Bastante útil se você tentar analisar uma fonte comprimida que possui partes descomprimidas como a do Phantasy Star II (que utiliza uma compressão RLE).

- *Editing fonts without the aid of a Graphics Editor!* (por Neil_): Não só te servirá para editar gráficos sem a ajuda de editor gráfico (algo que pode ser muito útil para copiar a fonte de um jogo para outro sempre que elas usarem o mesmo modo gráfico e paleta), todavia que explica no que se baseia um tile.
- *Title Screen Hacking Made Easy* (por InVerse): Se está interessado em trocar os gráficos da tela de apresentação, este documento pode ser útil.

CAPÍTULO 5:
PROCURA POR TEXTOS COMPRIMIDOS

É possível que com o método normal de procura que vimos no segundo capítulo deste manual não encontremos nada. O que é normal, sobre tudo em jogos programados pela companhia Squaresoft, pois o texto está comprimido. Adiante veremos o que podemos fazer quando se está diante desta situação.

5.1. Programas necessários e outros requerimentos

- *Martial*: Programa que se encarrega de procurar a tabela DTE ou MTE mais apropriada para nossos scripts.
- *O arquivo que se deseja traduzir*. Neste caso tomaremos como exemplo a ROM do Final Fantasy III(USA).

5.2. Compressão DTE e MTE

5.2.1. Averiguar a compressão DTE e MTE

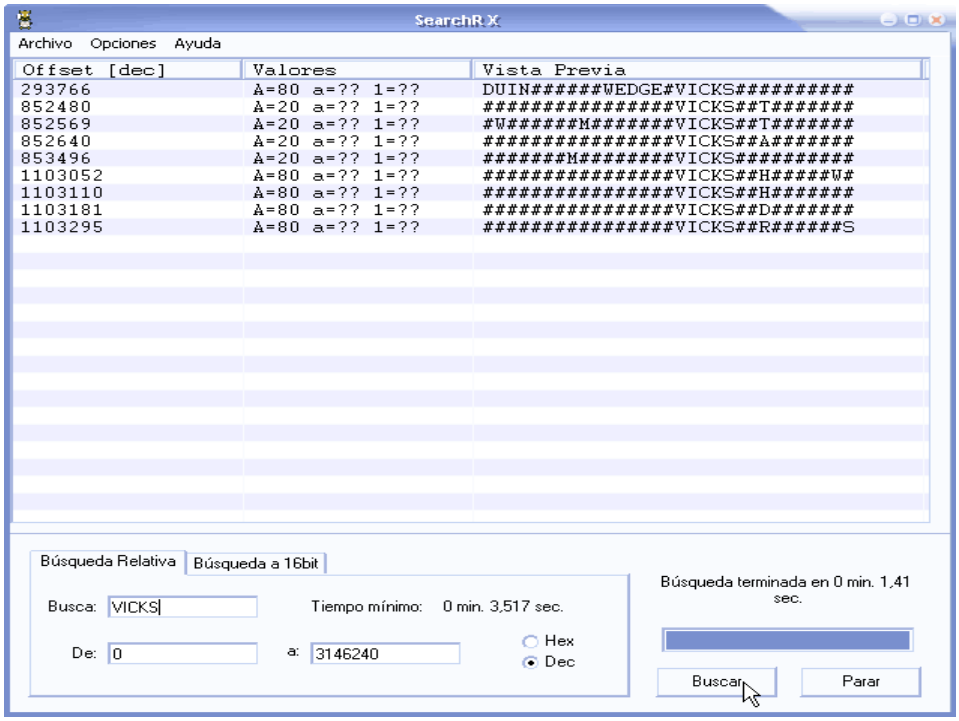
Muitos jogos de SNES empregam as técnicas DTE (Dual Tile Encoding) e MTE (Multi Tile Encoding) para comprimir o texto. Um código DTE comprime em um byte dois caracteres (daí o seu nome), pelo que, supondo, a palavra "Olá!" ocuparia dois bytes em vez de quatro. "Ol" ocuparia um byte e "a!" outro. Um código MTE é igual a um DTE coma diferença de que se pode comprimir mais de dois caracteres em um só byte, ainda que às vezes o número de caracteres comprimidos é tão grande que são codificados em dois bytes. Um bom exemplo disto podemos encontrar na ROM de Lufia de SNES.

Agora veremos como decifrar um texto que está comprimido por meio de DTE, já que é mais simples aprender com DTE pois ainda que o texto comprimido com MTE se baseie nos mesmo princípios, fazer uma tabela com códigos MTE é mais pesado e demorado. A ROM de Final Fantasy III de SNES (versão USA) utiliza esse tipo de compressão, deste modo o primeiro passo será executar o SearchR X e carregar a ROM. O que temos que fazer é procurar silabas fáceis, como pode ser "cas", "ca" ou "lo". Se não encontrarmos nada, deveremos tentar com outras silabas que nos vier em mente (claro, devem ser parte de uma palavra inglesa). Com um pouco de sorte logo encontraremos algo.

Para termos certeza de que são os valores que procuramos, podemos editar o texto que aparece e ver se há alguma coisa diferente quando jogamos. Para isso, devemos olhar em que posição está o texto encontrado e assim ir até ela com o Translhextion por meio da função *Jump* to. Este método possui o inconveniente de que pode levar muito tempo, mas se o texto está com esse tipo de compressão é a tarefa mais fácil para se dar cabo dela.

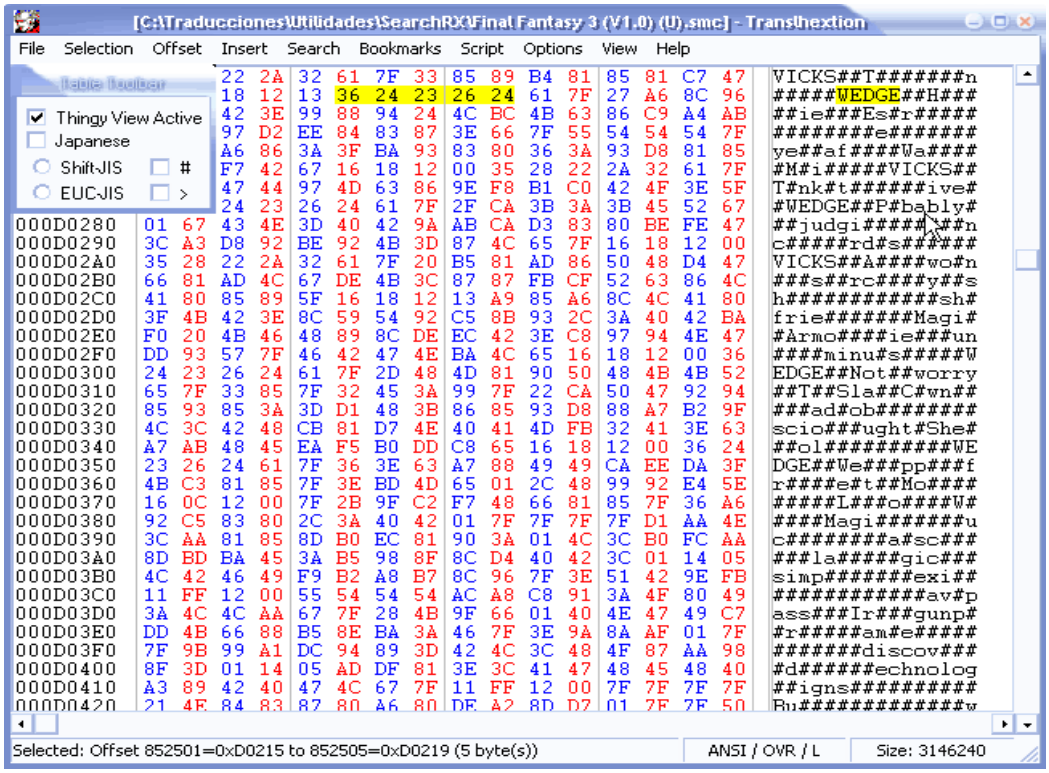
Porém, há um método mas fácil e quase sempre funciona, que é aproveitar-se do texto que está em maiúscula. O que quer dizer isso? Que devido a sua pouca freqüência nos textos é mais provável que todas as palavras que estejam em maiúsculas não estejam comprimidas. O inconveniente é que devemos recordar de um texto que apareça em maiúsculo e que se possa visualizar no emulador para verificar as mudanças.

No Final Fantasy III (USA) os nomes das personagens não protagonistas estão em maiúsculas.Além disso, existe um diálogo justamente no inicio, que trás a tradução, onde aparecem dois desse nomes em maiúsculas: VICKS e WEDGE. Por tanto, o que é preciso fazer é executar o SearchR X para procurar por VICKS:



A equivalência do primeiro resultado tem toda a pinta de ser a tabela dos menus porque está junto a outros nomes, mas fazer e completar tal tabela não é o objetivo deste trabalho. Se nos fixarmos nas equivalências do resto dos resultados, todos são 20=A. Ainda não se vê texto ao lado de VICK, resultado estranho que sempre mostra a mesma equivalência.

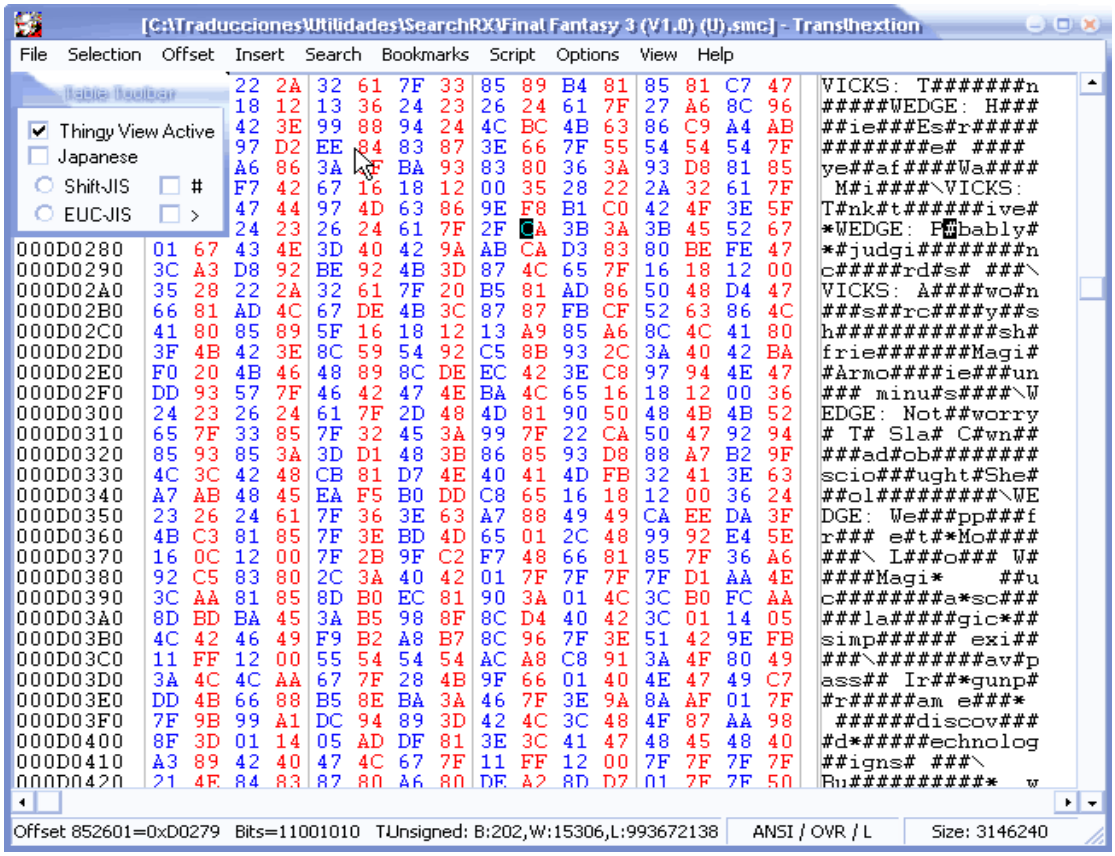
Agora devemos fazer a tabela das maiúsculas (lembra que 20 equivale a 'A') tal como já explicamos no segundo capítulo fazendo uso do TaBuLar. Como também já comprovamos antes, justamente depois de 'Z' maiúsculas vem o 'a' minúsculo. Assim, depois do valor de 'Z' clicamos em *Insert Lowecase Alphabet* nos menu *Insert*. Feito isso, salve a tabela e voltamos para o Translhextion. Carregamos a ROM de Final Fantasy III (USA) e tabela que acabamos de fazer e damos um *Find using Table* nos menu *Search*. Se procurarmos por WEDGE no lugar de VICKS, nós encontraremos o que pode ser visto na seguinte imagem:



Parece que, efetivamente, depois de 'Z' vem o 'a', porque somos capazes de ver alguns textos legíveis entre tantos #. Se não fosse assim, seria questão de substituir alguns códigos por outros e verificar o resultado jogando o jogo no emulador para ver o que aparece na tela.

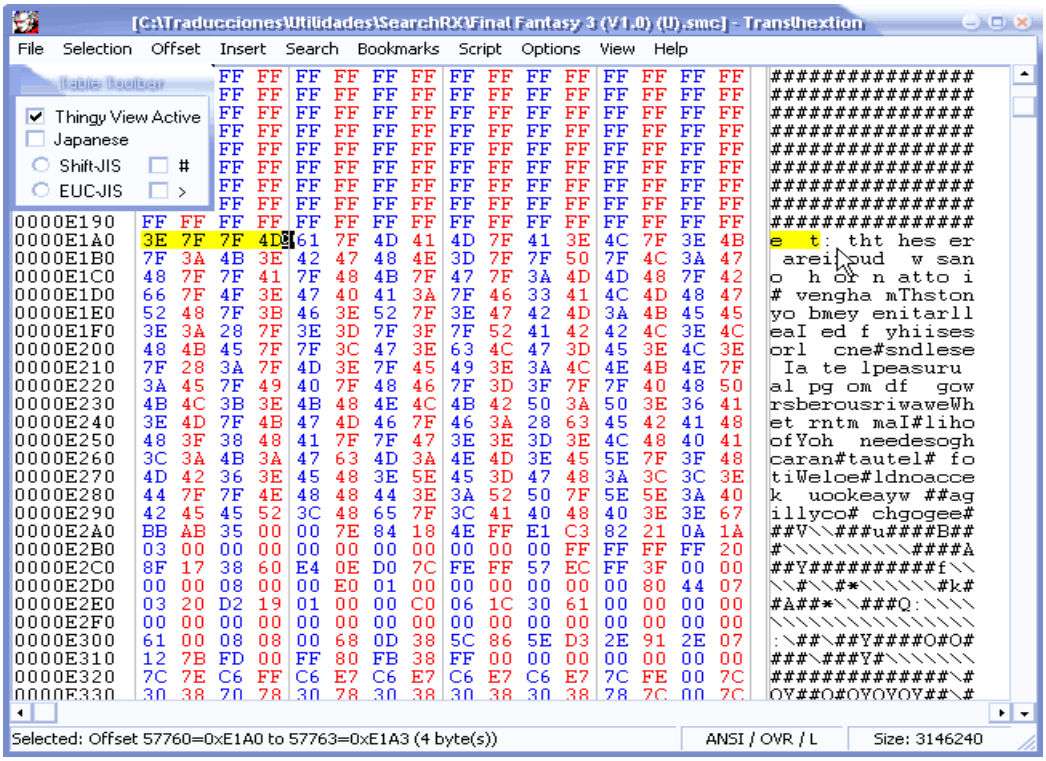
Já demos um grande passo, pois encontramos o texto, embora agora venha o que não é difícil, mas que leva tempo: completar a tabela. É preciso ter em mente que um código hexadecimal equivale a dois caracteres e não a um, devido a compressão DTE, embora ainda nos falte colocar os sinais de pontuação, não podemos esquecer que um # também pode equivaler a um só caractere. Como sabemos que depois de cada nome de personagem há dois ponto (:) e um espaço, é fácil verificar tais valores. Basta ir até onde esses valores estariam, e acrescentá-los à tabela.

Logo perceberemos que o código de quebra de linha é o 01 e o de fim de caixa de texto é o 00. De fato, o de quebra de linha é tirado por dedução após analisar o fim da caixa de texto, já que é bastante freqüente em uma ROM que 00 seja o código de fim de texto e 01 o de quebra de linha. Uma vez encontrados e colocados todos esses valores na tabela será mais fácil deduzir os valores das DTEs, então voltamos a carregar a tabela com os novos valores acrescentados, nós encontraremos isso:



O que devemos fazer é ir pouco a pouco encontrando os #. O código DTE que está justamente onde está o cursor na imagem tem a pinta de corresponder a “ro”, olhamos seu valor hexadecimal e o acrescentamos na tabela da seguinte maneira: CA=ro. Para começar a traduzir só nos resta completar a tabela com os valores dos códigos DTE. Por ser uma tarefa um pouco árdua recomendo por em prática o truque de colocar uma sucessão de códigos hexadecimais para averiguar seus valores como vimos no terceiro capítulo. Porém recomendo por um espaço (neste caso o 7F) entre código e código porque pode ficar difícil ler tantos caracteres juntos e por tanto averiguar quais caracteres correspondem a cada código DTE.

Sem problema, o jogo deve ter em algum canto a lista de compressão com todos os caracteres comprimidos, bem juntos ou bem separados por algum código. O que temos que fazer é procurar os valores dos códigos de forma seqüêncial usando uma tabela que não contenha nenhum valor DTE ou MTE. Por tanto, se possuirmos os valores de, por exemplo 80 (“e”) e 81 (“t”), vamos ao inicio do Translhextion, carregamos a tabela sem nenhum valor DTE e lhe damos um *Find using Table*. Se procurarmos “e t” acharemos o seguinte:



Por sorte encontramos o que procurávamos. Simplesmente devemos fazer a tabela da seguinte maneira:

80= e

81= t

82=:

83=th

84=t

85=he

...e assim sucessivamente. Ficando com o endereço da posição do local em que começa esta lista de compressão, chamada tabela de DTE e MTEs (usar um *Bookmark* do Translhextion é mais que recomendado), porque precisaremos dela mais tarde.

5.2.2. Alterar a compressão DTE e MTE original

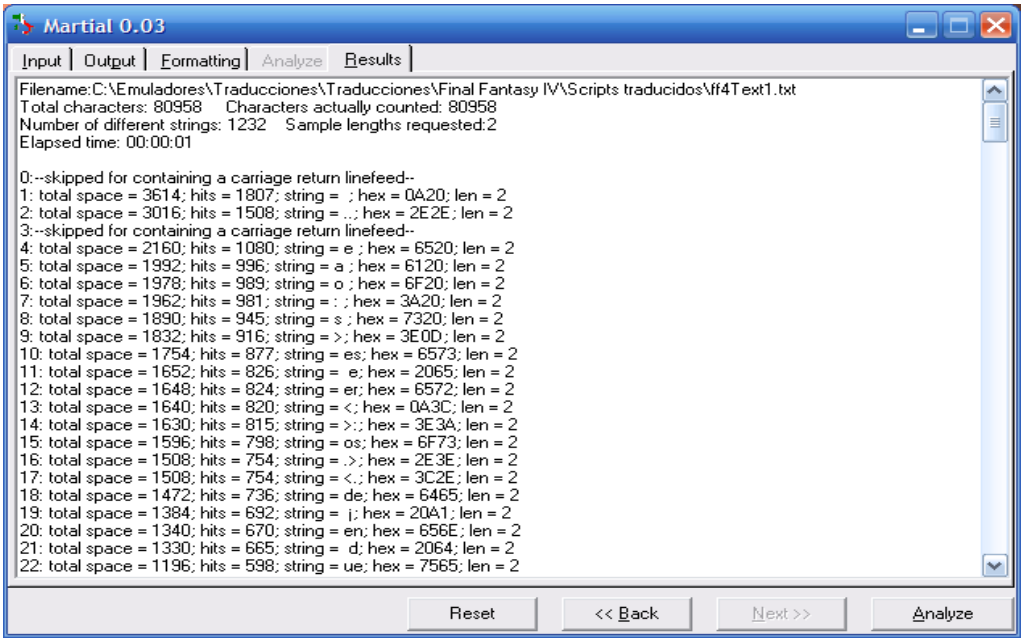
Agora veremos como podemos nos aproveitar da compressão para tornar a tradução um pouco mais fácil. Lembre-se que isto o obrigará a ter duas tabelas (a original e a modificada com as trocas que fizemos) e a jogar uma ROM diferente da qual se está traduzindo para ler o texto original porque este ficará ilegível após trocar as DTE e MTEs. Tudo isso será desnecessário se optarmos por usar scripts como veremos no próximo capítulo deste manual.

O que temos que fazer é trocar a tabela DTE e MTE original, porque, por exemplo, trocaremos “st” por “si”, sílaba que iremos utilizar muito em nossa tradução. Para isso, devemos ir ao local onde estavam todos as DTE juntas como foi visto anteriormente. O que é preciso fazer é substituir “st” por “si” e salvarmos a troca. A partir deste momento, em vez de aparecer “st” irá aparecer “si” na tela. Devemos repetir esse procedimento até que não reste mais valores de DTE. Enquanto realizar essas trocas, é mais que recomendado fazer uma tabela com os novos valores e não deixá-lo para o final.

Isso levará o texto da ROM aparecer ilegível, daí a necessidade de ir jogando uma ROM sem modificações para saber o que aparecerá realmente na tela. Para traduzir você pode usar o Thingyv, já que ele permite carregar duas tabelas por vez. Basta usar a tecla TAB para mudar entre uma e outra.

Se trabalharmos com scripts nos será muito mais fácil optimizar uma tabela de DTE ou MTE de acordo com os scripts traduzidos. Ainda que tudo relacionado com scripts será tratado no capítulo 7, creio que seja conveniente explicar isso neste momento.

O programa Martial se encarrega de analisar as repetições mais freqüentes de dois ou mais caracteres juntos por todo o script. Vejamos como optimizar um script mediante este programa. Na guia *Input* devemos especificar o script ou scripts que queremos analisar, além do comprimento dos caracteres que se vai procurar. Como queremos buscar DTEs temos que designar i, valor de 2 na caixa de texto de Sample *Length(s)*. Passemos para a guia *Output* indicaremos um valor de 300 em *Only Show Top X Results*. Este alto valor justifica muitos das possíveis DTEs não nos servirão por diversos motivos, como que haja um caractere próprio do script (por exemplo, <o>) na possível DTE. Nesta guia devemos especificar os arquivos onde serão salvas as resultados e a nova tabela DTE, algo que nos poupará um esforço considerável. Na terceira guia ativaremos as opções *Ignore Carriage Return-Line Feeds*; uma vez tudo pronto poderemos clicar o botão Analyze. Depois de alguns istantes, aparecerão todos os resultados na tela:



Agora é só dar uma olhada nos resultados e modificar na tabela gerada aqueles valores que nos façam falta, tais como (.<) de acordo com os resultados que aparecem na tela. Lembre-se que o ponto-e-vírgula que aparece atrás de cada valor não é colocado na tabela.

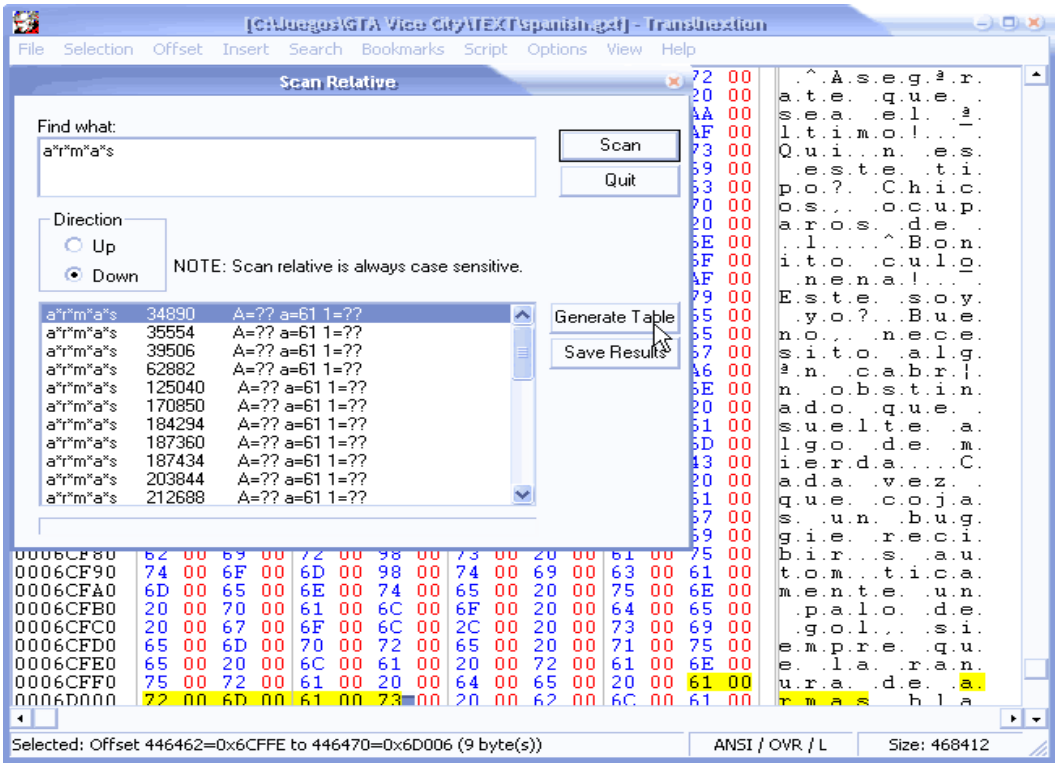
5.3. Procurar texto em modo 16 bits

Que o texto se encontra em modo 16 bits não significa realmente que está comprimido, sendo que um caractere ocupa dois bytes em vez de um, ao contrário da compressão DTE. Sinceramente, desconheço a razão disto. Normalmente nestes texto existe um código "normal" junto a outro que sempre se repite (geralmente 00).

Não existem muito jogos com o texto codificado em 16 bits, ainda que, supostamente o Super Metroid de SNES seja um deles. O Zelda de SNES usa este modo para a descrição dos objetos se me lembro bem. Em qualquer caso, é utilizado por um jogo de PC, o Grand Thelf Auto 2. Os jogos de PC só possuem este modo ou igual que usam uma tabela ASCII.

É muito fácil localizar o arquivos que contém os textos deste jogo, já que a pasta TEXT e o arquivo que se chama spanish.gxt dentro dela dizem tudo. Desta vez vamos usar o Translhextion para procurar o texto, porque o SearchR X é muito lento e na mostra direito os resultados que se procura em modo 16 bits.

Dito isso, executamos o Translhextion e abrimos o arquivo mencionado. No *Search Relative* clicamos em *Search* e digitamos, por exemplo, a*r*m*a*s. O asterisco atua como código comum quando se procura o texto; esse será o código que se repete como já foi dito no início. Não precisa que façamos a tabela com o TaBuLar desta vez, já que podemos salvar a tabela diretamente com o Translhextion. Espero que tudo fique mais claro com esta imagem.



Como não incluímos nenhuma letra maiúscula na procura, só serão salvos os valores das minúsculas. No caso de querer acrescentar os valores das maiúsculas, simplesmente procure por uma palavra que possua uma. Não deve esquecer que, como o código que sempre se repete é 00, terá que acrescentar 00 a cada valor, por exemplo, o 'a' não deve aparecer na tabela como 61=a, e sim como 6100=a.

5.4. Para saber mais

Infelizmente para nós romhackers e felizmente para os programadores, não existe apenas texto comprimidos com DTE e MTE. Se não encontrar o texto o mais certo é que o esteja comprimido por meio dos algoritmos Huffmam ou LZ77 (ou variantes deste) ou outros. Eu nunca trabalhei com nenhum desses tipos de compressão, de modo que não posso oferecer muitos detalhes. Por tanto, proponho ler uns documentos muitos interessantes que podem servir de guia, ainda que, lembrando que não é algo fácil e que requer tempo para compreender a compressão utilizada no jogo. Além disso, em alguns caso necessitará que saiba programar para elaborar uma aplicação que descomprima e volte a comprimir os textos.

- *Comprendre la Compression de Huffman* (por Ti Dragon y Meradrin): Se até agora não conseguiu muito com o que foi explicado sobre a compressão Huffman, porventura encontrará uma explicação melhor neste tutorial.
- *Compression Tutorial* (por Jay): Explica detalhadamente os diferentes tipos de compressão que se pode encontrar. Muito recomendado.
- *Descompresión del Intro del Secret of Mana* (por Dark-N): Excelente guia que explica detalhadamente passo a passo como funciona a compressão LZSS (ou melhor uma variante desta) da introdução do Secret of Mana. Todavia, existem algumas partes que são um pouco obscuras até para o autor.
- *Entender los Algoritmos de Compresión de Huffman y LZ* (por Dark-N): Um estupendo documento em espanhol que fala sobre estas compressões. Também explicam com exemplos para favorecer a compressão do documento.
- *Huffman Decoder* (creio que seja Bongo): Um programa escrito em C para descomprimir textos que se encontram com esta compressão, ainda que supondo que varia de acordo com o jogo. Inclui o código fonte em C e também uma interessante rotina em *assembly* de SNES.

CAPÍTULO 6:
PONTEIROS

Neste capítulo explicarei tudo relacionado com os ponteiros, se bem que será dito nada relacionado sobre a modificação destes. Normalmente modifica-se quando utilizamos scripts para traduzir, o que foi preferido deixar a explicação para o próximo capítulo.

6.1. Programas necessário e outros requerimentos

- *WindHex32*: editor hexadecimal alternativo ao Translhextion. Durante o capítulo veremos porque utilizá-lo.
- *Lion Pointer Calculator*: ainda não foi mencionado neste manual, este programa será muito útil para calcular ponteiros de diferente sistemas (consoles).
- *Arquivo que tenha ponteiros*. A modo de exemplo utilizaremos a ROM de Final Fantasy III(USA) de SNES, ainda que será mencionado outros jogos.

6.2. Introdução aos ponteiros

Um ponteiro é um endereço que indica desde onde se deve começar a ler o texto de um jogo. O jogo deixa de ler o texto quando encontrar um código de fim de mensagem; esse código é que se pode encontrar ao final de um diálogo, normalmente representado pelos códigos hexadecimais 00 ou FF.

Se já tentou trocar o lugar do código de fim de mensagem, com muita sorte terá conseguido mudar o espaço original dos diálogos, quero dizer, a quantidade máxima de caracteres que podem ser usados em um diálogo. Isso acontece na ROM de Zelda de SNES, mas infelizmente se trata de uma exceção à regra. A resposta para a pergunta de qual é a utilidade dos ponteiros é bem simples: alterando os ponteiros podemos utilizar mais texto que o original. Tentarei explicar seu funcionamento com o seguinte exemplo:

Welcome.#We have some problems with that new king.#Hello, my name is Lorena.#

Se optarmos pelo método de trocar o código de fim de mensagem (o # que aparece depois de cada ponto), trocando somente a primeira oração teríamos:

Bem-vindo.#have some problems with that new king.#Hello, my name is Lorena.#

Se tivermos um jogo, com esse texto e o modificarmos tal como se vê neste segundo texto, comprovaríamos que onde deveria aparecer *Welcome* agora se mostra *Bem-vindo*. Sem problema, a segunda oração não mostrará *have some problem with that new king.*, sendo simplesmente "*do*." Isso acontece porque há um ponteiro que aponta (daí o nome de ponteiro) ao caractere do começo do segundo diálogo. Por tanto, alterar o tamanho de um diálogo sem alterar o resto será necessário alterar os ponteiros e não o código de fim de mensagem.

6.3. Como encontrar a tabela de ponteiros

Uma tabela de ponteiros refere-se a um lugar do arquivo em que se encontram todos os ponteiros. Estes estão normalmente de forma ordenada, ou seja, que o primeiro ponteiro corresponde ao primeiro diálogo segundo aparece o texto internamente no arquivo, o segundo ponteiro ao segundo diálogo e assim sucessivamente. Existe uma forma padrão para verificar isso, ainda que com o tempo espero que não seja necessária a técnica que proponho adiante. Novamente, tomaremos como exemplo a ROM do Final Fantasy III (USA). Tenho que dizer que os ponteiros são geralmente de dois bytes, mas existem de três e até de quatro (o Suikoden de PSX e o Phantasy Star IV de Megadrive utilizam ponteiros de quatro bytes).

Primeiramente devemos carregar a ROM no Translhextion e ir ao primeiro caractere do primeiro diálogo que existe na ROM (não tem que ser necessariamente o primeiro que aparece no jogo), que neste caso é "VICKS: There's the town...". Se nos fixarmos na posição do primeiro caractere (o 'V') veremos que está situado na posição 0D0200 em hexadecimal. Agora temos que aplicar a seguinte fórmula:

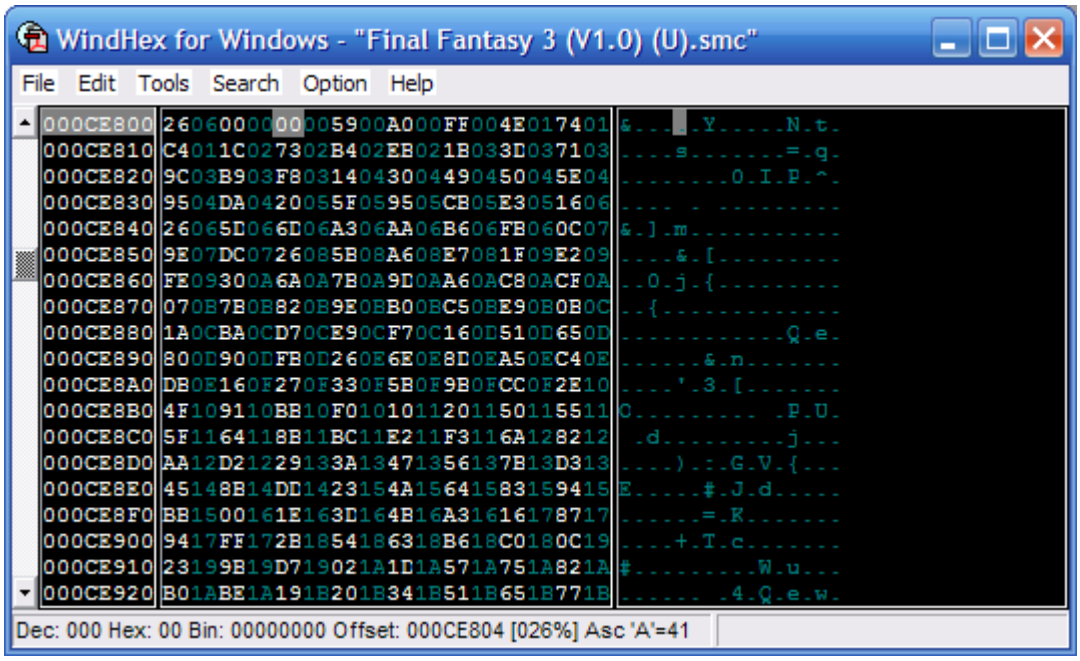
Ponteiro do diálogo X=(Posição do primeiro caractere do diálogo)-(header da ROM se existir)->(deixar os últimos quatros caracteres)->(agrupados em pares)->(invertidos).

Vejamos por partes para vê-lo de uma maneira mais clara:

1. O header de uma ROM (se possuir) pode variar segundo o seu formato, ou seja, uma ROM de SNES possui 512 bytes, igual a uma ROM de Megadrive, mas uma ROM de NES no formato .nes possui um tamanho de 16 bytes. Sigamos como o de antes. A posição do primeiro caractere do diálogo é, como verificamos antes 0D0200.

2. A 0D0200 ficamos com 200 neste caso porque 512 (o número de bytes do header de uma ROM de SNES) é 200 em hexadecimal. Desta maneira, obteremos 0D0000.
3. Deixamos os últimos quatro dígitos, pois de 0D0000 pegamos 0000.
4. Agrupamos os últimos quatro dígitos em pares, ou seja, 0000 passa a 00 e 00.
5. Por último, invertemos os dígitos; ao ser os dois códigos hexadecimais iguais não importa, mas se tivéssemos 01 45 esta seqüência numérica passaria a ser 45 01. É preciso fazer esse passo porque o SNES e outros consoles lêem os ponteiros ao contrario (para explicá-lo de uma maneira bem simples).
6. Agora o que é preciso fazer, é ir até o início da ROM para procurar esses valores. Antes de fazê-lo, repetiremos o processo com o diálogo seguinte, porque 00 00 é algo muito comum em qualquer ROM. Na realidade, quase todos os jogos possuem esse valores como primeiro ponteiro. Volte a repetir o processo com o local do diálogo seguinte, nos daremos conta de que 69 00 é o ponteiro do segundo diálogo.

Agora vamos ao Translhextion, carregamos a ROM e vamos a *Find* no menu *Search* com a opção *Hex* marcada. O que devemos fazer é procurar 00005900, mas então veremos uma das falhas que o Translhextion possui. Este não procura corretamente quando é acrescentado vários 00 no início da cadeia de busca. Como resultado disto deveremos usar um editor hexadecimal distinto, como por exemplo o WindHex32. Vamos em *Hex Search* no menu *Seach* e em seguida estaremos no local onde é mostrada na figura:



Tudo aponta que estamos diante da tabela de ponteiros. De fato, os resto dos códigos são muito parecidos, ou seja, que respeitam a ordem que se deve seguir nos ponteiros: os códigos da esquerda não váriam de ponteiro um a outro como faz o código da direita e aumenta em um quando o código da direita supera o valor de FF. Por outro lado, parece que o jogo possui uma rotina especial para detectar se o código da esquerda passa de FF, já que volta a 00, mas na realidade é como se carrega 01 00 00 (como se fosse um ponteiro de três bytes).

6.4. Outras formas de achar ponteiros

Se mediante o método explicado não se encontrar os ponteiros, podem haver várias razões. As seguintes são as diferentes possibilidades que eu encontrei ao longo de vários anos dedicados ao ROMHacking:

- **Que os ponteiros não sejam de 2 bytes**. Deixamos só os dois últimos códigos de cada posição. Isto se deve a que, normalmente, a maioria dos jogos usam ponteiros deste tipo. Sem embargo, outros jogos, como Final Fantasy V de SNES, utilizam ponteiros de 3 bytes. Outros, como o Suikoden de PSX, utilizam os ponteiros de 4 bytes. Isso só facilita a tarefa de utilizar textos de diferentes partes do arquivo com o que trabalharemos. Ainda existem ponteiros de um só byte como os que se encontram no Phantasy Star II de Mega Drive. Este utiliza uma técnica baseada em somar ao ponteiro a direção que se utiliza no diálogo prévio ao que o ponteiro em questão aponta. Por último, resta dizer que se nos encontrarmos com ponteiros de 3 bytes em vez de deixar só os últimos quatro dígitos, devemos deixar seis, assim como oito no caso de que se trate de ponteiro de 4 bytes.
- **Que os ponteiros estejam espalhados pelo arquivo e não em uma tabela de ponteiro**. No momento só os vi na ROM de Phantasy Star IV. Ao se usar ponteiros de 4 bytes, pode-se colocar o texto onde se queira na ROM. O único modo de os achar é procurar a direção de cada diálogo pela Rom e comprovar se realmente é um ponteiro o que encontrarmos. Por exemplo, se temos um diálogo que começa na direção 004588AB, devemos procurar a mesma seqüência na ROM (lembre-se que é possível que seja necessário subtrair os dois bytes do header da ROM se esta possuir uma) ou só a metade (88AB).

- **Que o jogo use uma maneira distinta de carregar o texto**. o Final Fantasy VI de PSX possui uma rotina que carrega algumas vezes textos do menu do objetos usando ponteiros e outras vezes não, ou ao menos é o que parece. Essa é a razão pela que decidi não traduzí-lo devido aos problemas que acarreta modificar os textos. Em Phantasy Star IV acontece algo parecido, pois há um suposto ponteiro nos objetos que eu nuca consegui encontrar. Em Phantasy Star III é utilizado também um sistema pouco comum de ponteiros que não descobri pois preciso trabalhar mais nele.
- **Que o jogo tenha um formato estranho de ponteiros**. Por exemplo, o Secret of Mana. No principio os ponteiros parecem seguir um padrão, mas logo percebemos que são diferentes e começam a não corresponder ao padrão. Além de, cada diálogo deste jogo esta rodeado de códigos. Magno do *Traducciones Magno* [http://www.nekein.com/magno/] deverá saber mais sobre isto, pois ele também traduziu o Secret of Mana.
- **Que a tabela de ponteiros não começar em 00 00**: como dissemos antes, a tabela de ponteiros geralmente começa por 00 00, mas outras vezes começa por outros valores diferentes (não digo arbitrariamente porque eles terão suas razões de ser). Teremos de ser espertos para encontrá-la; se soubermos assembly (quero dizer, conhecer e interpretar as operações do processador do jogo com o qual trabalhamos) e analisarmos os códigos do jogo, tarde ou cedo chegaremos na tabela dos ponteiros. Se não, que é o mais provável, o melhor é começar a examinar próximo do bloco de texto, já que a tabela pode estar perto deste, normalmente um pouco mais acima, ainda que às vezes esteja embaixo do bloco de texto.
- **Que a tabela de ponteiros possua ponteiros desordenados**. Pode ocorrer que encontre a tabela, mas que dá a impressão de não ser, porque todos os ponteiros estão desordenados (quero dizer, que não seguem a ordem do texto que há o arquivo). Comprove os ponteiros vendo se correspondem com seus endereços. Ainda que nunca topei com nenhum jogo que utilize tais ponteiros, eu li que existem alguns nesse formato.

- **Que simplesmente não haja ponteiros**. Pelo melhor que a principio aparenta ser, é improvável o paradoxo de sua verdadeira causa. Nunca é demais trocar o código de fim de mensagem para passar o texto e ver o que se mostra no dialogo "modificado", ou seja, o seguinte ao código de fim de mensagem. Isto acontece no Zelda de SNES e freqüentemente nos objetos (a parte de magias, habilidades, técnicas...) de jogos como o Phantasy Star II e IV. Algumas vezes nem sequer há o código de fim de mensagem , outras sim; a questão é que existe um número de caracteres fixos em todos os objetos, se ocupam ou não todos os caracteres disponíveis, como acontece no Final Fantasy III (USA) de SNES. Essa técnica é conhecida como *Constant Lenght Text*(Texto de Comprimento Constante). Um exemplo desta pode ser a seguinte, no que existeum máximo de oito caracteres por arma: *Sword###Spear###Bow####*.

6.5. Para saber mais

- *AnusP's Advanced Hacking Tutorial* (por AnusP): Outro documento muito interessante sobre ponteiros. Na realidade, eu aprendi com este.
- *Comment trouver les pointeurs 24bits SNES* (por Elfe Noire): O próprio titulo diz tudo. Nunca é demais saber um pouco sobre este tipo de ponteiros, pois você pode topar com ele.
- *The Mad Hacker's Guide to Pointers* (por The Mad Hacker): Interessante, completo e útil documento que explica detalhadamente como encontrar os ponteiros e quais tipos podem ser encontrados normalmente em uma ROM. Está baseado em ROMs de NES.
- *The Madhacker's Guide to NES Pointers: Appendix B* (por Gil_Galad): Explicação mais técnica e complementar ao que é explicado no documento original do MadHacker.
- *Using Pointers Effectively* (por Ghideon Zhi): Ainda que não seja descrito nesta parte do manual a modificação de ponteiros, que será visto no próximo capítulo, pode ser recomendado lê-lo para conhecer a maneira de aumentar espaço na ROM usando ponteiros.

CAPÍTULO 7:
SCRIPTS

7.1. Programas necessários

- *Vrecalc*: Um simples mas prático utilitário de MS-DOS que é capaz de economizar muito tempo e esforço recalculando para nós os ponteiros.
- *Tabela e ROM de Final Fantasy IV traduzida pela J2e*: a tabela é realmente tediosa de completar, por isso a inclui junto com esse manual. A ROM será por sua conta conseguir.
- *PSPad*: Excelente e gratuito editor de textos para Windows. Ainda que não o menciono em qualquer parte deste capítulo, o recomendo para traduzir scripts.

7.2. Scripts

7.2.1. Definição

Um script - termo em inglês para se referir aos textos de um jogo - é um arquivo de texto em que se encontra parte (ou todo) o texto do jogo, para ser traduzido mais comodamente com qualquer editor de texto. Este arquivo de texto será criado depois de extrair o texto do próprio arquivo que o contém e voltará a inserí-lo mais tarde no mesmo para que, a tradução deste modo seja aplicada no arquivo em questão. Sem problemas, os usos mais comuns que se dão aos scripts (aparte da flexibilidade que oferecem), são os de trocar mais tarde os ponteiros para inserir os textos e ter um texto com mais qualidade e dispor de mais espaço que o texto original.

7.2.2. Vantagens e inconvenientes

Evidentemente, nem tudo são rosas. Vejamos o que nos oferecem os scripts. Por um lado temos comodidade flexibilidade na hora de traduzir o texto. Mas traduzir o texto sem jogar, pode levar a traduzir coisas indevidamente como se viu no primeiro capítulo deste manual, pois não são raras as vezes que é difícil distinguir o significado de uma palavra ou expressão em concreto ou saber se “you” se utiliza em singular ou plural. Em outras palavras, quem não conhece o contexto da mensagem, cuja importância ficou clara no primeiro capítulo.

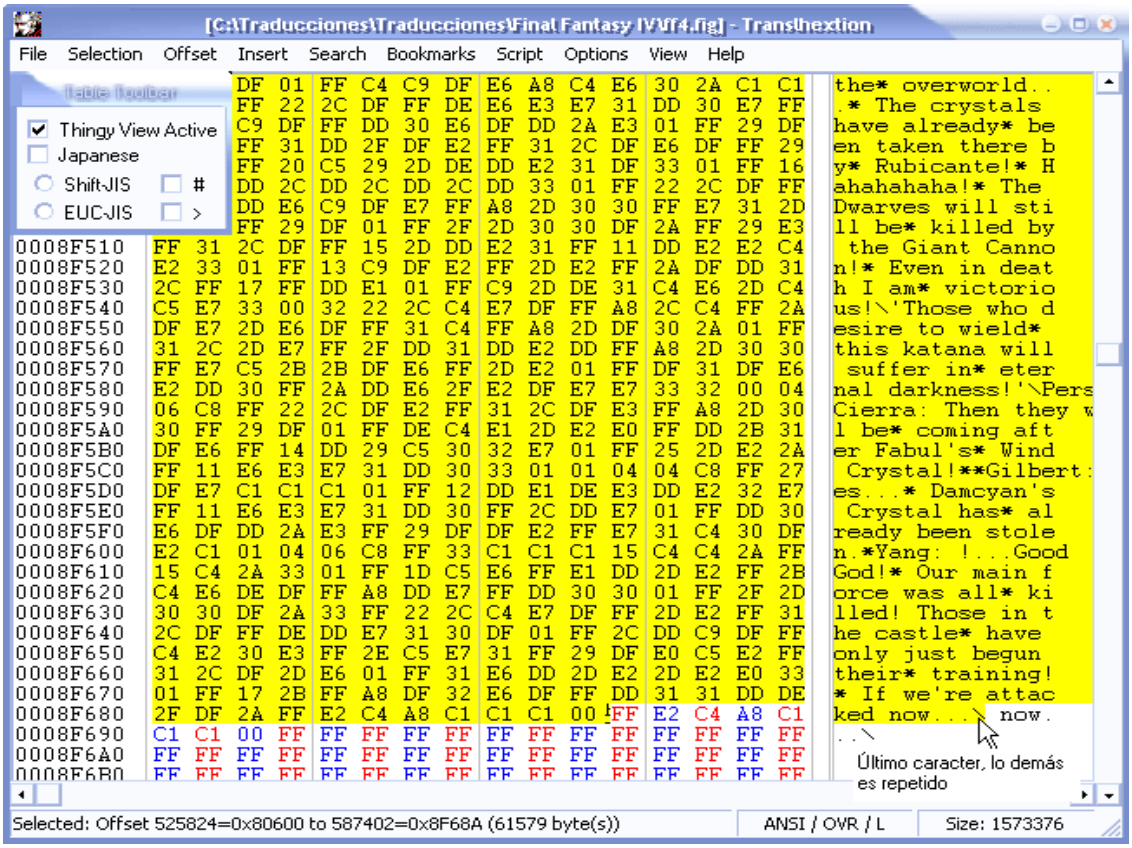
Também pode ser tedioso inserir os scripts e recalcular ponteiros. De toda maneira, a pesar de tudo isto, é muito melhor utilizar os scripts para ter uma melhor qualidade de texto (ao se recalcular ponteiros, o espaço original aumenta) e mais comunidade se o jogo for traduzido por mais de uma pessoa. Temos que lembrar que o tradutor deve ter todas as facilidades possíveis, pois traduzir algo cheio de códigos estranhos entre os textos e que por sua vez tenha que respeitar os espaço original tira a vontade de qualquer um.

7.3. Extrair o texto

Vamos usar como exemplo a ROM do Final Fantasy IV traduzida pela J2e. O primeiro que devemos fazer é rodar o Translhextion. Carregamos a ROM e a tabela correspondente e procuramos o primeiro código que aparecer no jogo:

Não apareceremos no primeiro diálogo do bloco de texto desse local da ROM pois este diálogo não é o primeiro que aparece internamente na ROM. Por tanto, retrocederemos até chegar no primeiro de todos, “Your inventory is full”. É recomendável criar o *bookmark* (no menu *Bookmark* do Translhextion) para mais tarde voltar ao local mais rapidamente. Se você compreendeu sem problema o que são as tabelas de ponteiros, poderá comprovar claramente que os ponteiros desse bloco de texto estão justamente acima.

No momento sabemos onde começa o texto. Agora o que temos a fazer é ir selecionando o texto mantendo pressionada a tecla Shift até encontrar o último texto de diálogo desse bloco de texto tal como se pode ver na imagem:



Nos encontramos no último caractere (que deve ser um fim de mensagem), e clicamos em *Dump Script* no menu *Script*. O quadradinho de *Separated-Byte format* não os recomendo que marque se seu jogo utilizar muita DTE/MTE porque pode trazer problemas na hora de inserir os script. Além disso, deixará praticamente uma tradução mais incômoda, porque todos os DTE e MTE terão < e > ao redor deles. Agora é só clicar em OK; se tudo está correndo bem, terá um bonito arquivo de texto com o texto inteiro ou parte dele do jogo com o que está trabalhando.

7.4. Inserir textos

Uma vez terminado de traduzir os scripts, ou simplesmente queremos comprovar como está nossa tradução da ROM, temos que inserí-los novamente na arquivo do qual os extraímos. A tarefa no início é bem simples ainda que possamos encontrar problemas se não a fizermos da maneira certa. A única coisa que devemos fazer é clicar em *Replace Script* no menu *Script* do Translhextion, escolher os mesmos parâmetros que escolhemos para extrair e clicar em OK. Se não tiver nada errado com os scripts, depois de alguns segundos o script estará reinserido no arquivo desejado, ainda que às vezes por motivos desconhecidos o programa possa dar erro. Em tal caso, seremos nós os responsáveis de encontrar o que causou a falha no script modificado.

7.5. Recalcular ponteiros

Pelo nome deste tópico há de se supor o mesmo deveria estar no capítulo dedicado aos ponteiros, creio que seja mais adequado explicá-lo agora. Espero que o leitor compartilhe da mesma opinião uma vez que tenha escolhido pela tradução de algum jogo que tenha que recalcular ponteiros.

Bem, uma vez que tenhamos inseridos os scripts traduzidos, copiamos a ROM modificada para a pasta onde esta o *Vrecalc*. Como o programa é de MS-DOS e é mais seguro que tenhamos que escrever os mesmo parâmetros uma ou outra vez, será melhor criar um arquivo de texto com a extensão .bat (quero dizer, não o salve como texto) e escrever em hexadecimal o que se diz na documentação do programa do Vegetal Gibber (todas as explicações são dele, não minhas, ainda que tenha adaptado algumas):

VRECALC(arquivo)(p.texto)(p.ponteiros)(f.ponteiros)(tam.ponteiros)(separador)[prim.ponteiro]

(arquivo) = Nome de arquivo da ROM.
(p.texto) = Posição inicial do texto.
(p.ponteiros) = Posição inicial da tabela de ponteiros.
(f.ponteiros) = Posição final da tabela de ponteiros.
(tam.ponteiros) = Tamanho em bytes de cada ponteiro (2-4).
(separador) = Valor hexadecimal do código de fim de mensagem.
[prim.ponteiro] = Valor do primeiro ponteiro (opcional). Se não se especificar se tomara como referência o valor original do primeiro ponteiro da tabela a ser recalculada.

Assim, em nosso arquivo .bat deveríamos escrever o seguinte:

VRECALC.EXE ff4.smc 80600 80200 805FF 2 00

Agora bastará executar o arquivo .bat dando um duplo clique sobre ele. Se foi feito tudo certo, teremos todos os ponteiros recalculados sem problema algum. Se não, revise o que pode ter falhado. Temos que ter o cuidado de que o texto inserido não utilize o mesmo código de fim de mensagem (por exemplo, 00) para outro propósito, como pode representar o nome de um personagem, por exemplo, 0002. Se for assim, o programa falhará por que ele não distingue entre o nome e o código de fim de mensagem.

7.6. Para saber mais

Mesmo não sendo propriamente documentos sobre scripts, mas sim sobre ROM Hacking em geral acredito ser oportuno estes tutoriais:

- *Guide de la Traduction* (compilada por la T.R.A.F.): Neste tutorial em francês são reunidos os documentos mais famosos escritos em francês sobre o ROM Hacking. E o mais interessante de todos é o de ASM do Skeud.
- *Romhacking Profissional* (por xxcunix). Documento em português, um pouco confuso mas que abrange muitos dos temas de ROM Hacking.

CAPÍTULO 7: SCRIPTS

CAPÍTULO 8:
A TRADUÇÃO DE JOGOS DE PSX

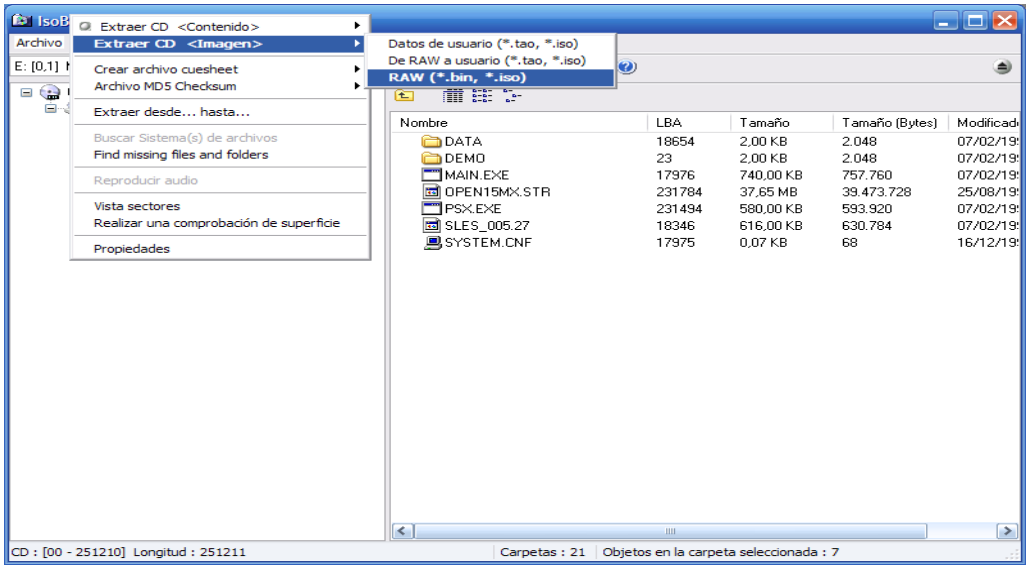
Decidi dedicar um capítulo especifico sobre a tradução de jogos de PSX porque seguindo o método "tradicional" nós encontraremos várias barreiras, que seja como encontrar os textos em um CD ou bem como modificar os gráficos. Muita gente segue empenhada em trabalhar sobre uma imagem ISO de um jogo de PSX (um arquivo que contém todos os dados do CD), ainda que eu tenha a opinião de que não há nada melhor que trabalhar com arquivos pequenos ainda que tenha que ter mais cuidado de saber em que arquivo está os dados que modificamos.

8.1. Programas necessário e outros requerimentos

- *ISOBuster*: Excelente programa para manejar imagens de CD. Será necessário para criar a imagem ISO do jogo que vamos traduzir.
- *CDMage*: Também permite manipular imagens de CD ainda que com mais limitações que o ISOBuster. Porém, possui a opção de poder substituir arquivos dentro de uma ISO, o que nos permitirá inserir os arquivos que traduzimos na ISO novamente se algum corromper os dados.
- *TIM Viewer*: Com este programa seremos capazes de manipular sem problema aqueles gráficos que se encontram no formato TIM.
- *ePSXe*: Emulador de PSX com o qual podemos comprovar nossos avanços graças à possibilidade de carregar imagens ISO.
- *O jogo para traduzir*: Se tomará como exemplo o Suikoden de PSX em sua versão PAL.

8.2. Criação de uma imagem ISO

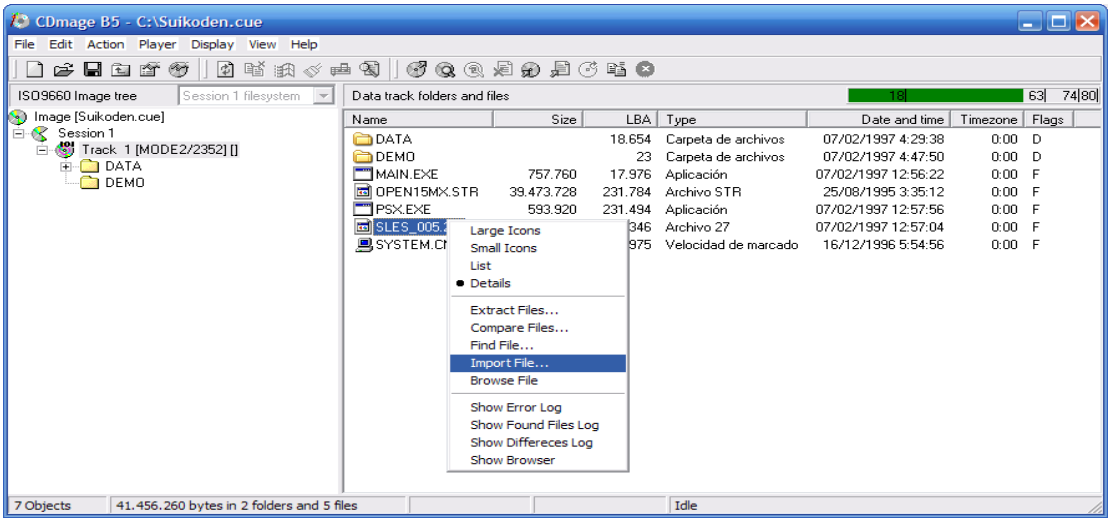
O primeiro que temos que fazer para trabalhar é criar a imagem ISO do jogo. Para isso, executamos o ISOBuster com o CD do Suikoden inserido em nosso leitor CD/DVD. O que temos que fazer é dar um clique com o botão direito sobre o ícone em que colocou o CD e escolher *Extrair Imagem>Raw* (*.bin, *iso). E continuando damos o nome ao arquivo que vamos criar (por exemplo, *Suikoden.bin*). Depois de vários minutos teremos em nosso HD a imagem ISO do Suikoden (ainda que se denomine imagem ISO, ela está no formato .bin). Quando o processo de extração for finalizado, ele nos perguntará o nome de um arquivo .cue; o mais recomendado é dar o mesmo nome que a imagem ISO.



8.3. Inserção de arquivos na ISO

Ainda que agora não possamos inserir nenhum arquivo modificado, simplesmente porque não os temos, acredito ser conveniente explicar neste momento a maneira de inserir arquivos modificados na ISO que criamos. Para isso devemos fazer uso da última versão beta do CDMage.

A tarefa a ser realizada é bem simples. Basta abrir o arquivo .cue que foi criado no processo de criação da imagem ISO, selecionar o arquivo que queremos substituir, dar um clique com o botão direito do mouse sobre ele e ir em *Import File*. Selecionamos o arquivo modificado de nosso HD e em alguns segundos as trocas nesse arquivo estarão também dentro da ISO, agora modificada. Devemos recordar que o arquivo a ser inserido deve ter o mesmo tamanho do original, já que assim os dados não serão perdidos, o que pode deixar inutilizada a ISO.



8.4. Procura de arquivos com o texto a traduzir

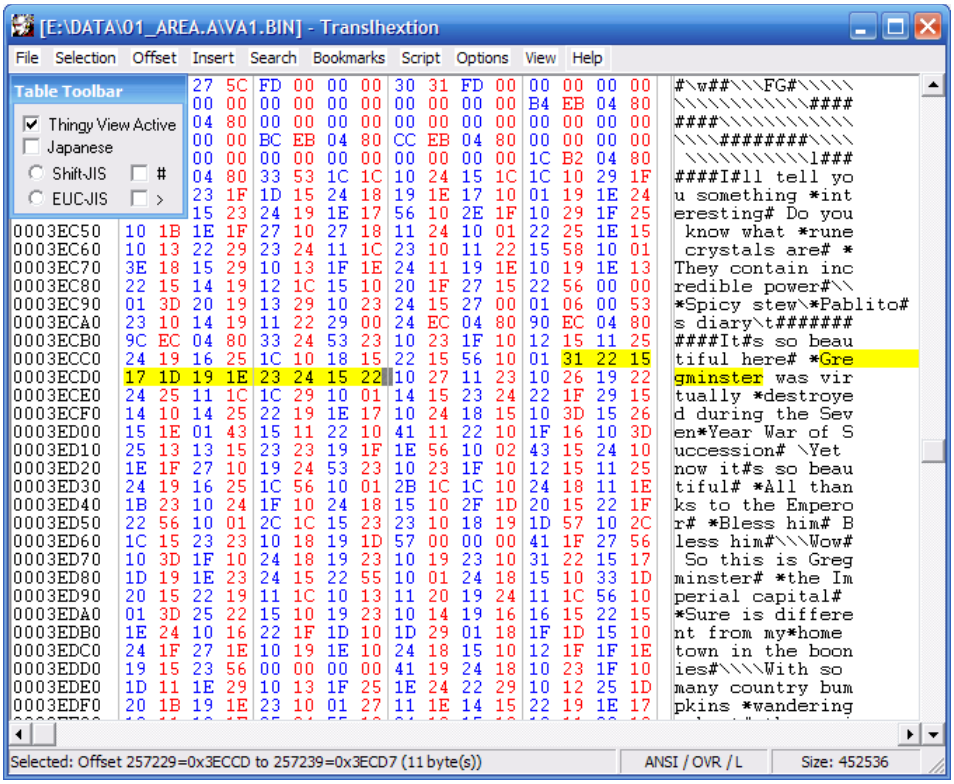
A maioria das pessoas agora explicaria como procurar o texto dentro da imagem ISO que acabamos de criar, mas eu proponho outra solução. O que devemos fazer agora é localizar os arquivos que contém os dados que nos interessam, ou seja, gráficos e textos. O Suikoden é um bom exemplo, pois possui uma quantidade notável de arquivos. É inevitável começar a procurar arquivo por arquivo ainda que fazendo uso do sentido comum podemos economizar muito tempo, sobre tudo porque o nome de um arquivo normalmente têm muito em comum com o seu conteúdo. De todas as formas, não confie sempre no conteúdo de um arquivo já que é possível que o texto se encontre repetido em mais de um deles.

O mais normal é que se encontrem todos os arquivos com os textos em uma mesma pasta e que, além disso, em casos de RPGs, os textos dos combates, dos menus e das conversas se encontram em um arquivo diferente aos dos diálogos do jogo. Se nos fixarmos na estrutura de arquivos do Suikoden, tudo aponta que o texto se encontra em alguma parte da pasta DATA. Além disso, podemos pensar de maneira lógica, e dado que há muitas pastas com o nome de [XX]_AREA.[Y] (onde XX é o número e Y uma letra), tudo parece indicar que o jogo carrega o texto de uma pasta diferente segundo o local em que nos encontramos jogando. Assim, a cidade de Greminster que é onde começamos, pertencerá à primeira, enquanto as outras cidades ou aldeias como dos anões correspondam às outras pastas.

Para procurar o texto usaremos o método explicado neste manual, ou seja, mediante o SearchR X. Basta carregar os arquivos diretamente do CD, ainda que seja preferível copiá-los para o HD porque antes ou depois teremos que fazê-lo. Teremos que descartar extensões de arquivos (ou simplesmente arquivos se prefere dizer assim) cuja função é nos deixar saber que não tem nada haver com o texto. Por tanto, devemos ignorar arquivos com extensão .str já que são arquivos de vídeos do PSX assim como a extensão .xa, que corresponde aos arquivos de som; Se nos aventurarmos na pasta 01_DATA.A veremos que existem extensões .va, .vb, .bin e .8. Como já dissemos que a primeira área deve conter os textos de Gregmunster vamos procurar “Gregminster” em todos os arquivos, não sem antes analisar as extensões que temos.

Ainda que muitos os desconheçam, os arquivos .va y .vb são arquivos de sons. Isso acontece porque há um programa que se encarrega de converter o som desses arquivos em formato .wav. Em caso de desconhecer, o único que haveria passado é que teríamos procurado por textos nestes arquivos e nos daríamos conta de que neles não se encontra nada.

O seguinte seria procurar nos arquivos .8, também sem resultado. Então só restaria os arquivos com extensão .bin que têm em toda a pinta de ter o que procuramos. Se procuramos "Gregminster" no arquivo VA1.bin encontraremos o texto que procuramos. O mesmo acontece se fizermos o dito no arquivos VA2.bin, já que além disso os valores para "a" e "A" são os mesmo em ambos os arquivos. Em boa hora, encontramos a chave: o textos de Suikoden se encontram nos arquivos .bin de cada pasta como nome AREA. Agora só falta criar uma tabela básica, abrir os arquivos em questão com o Translhextion e comprovar que estávamos certo.



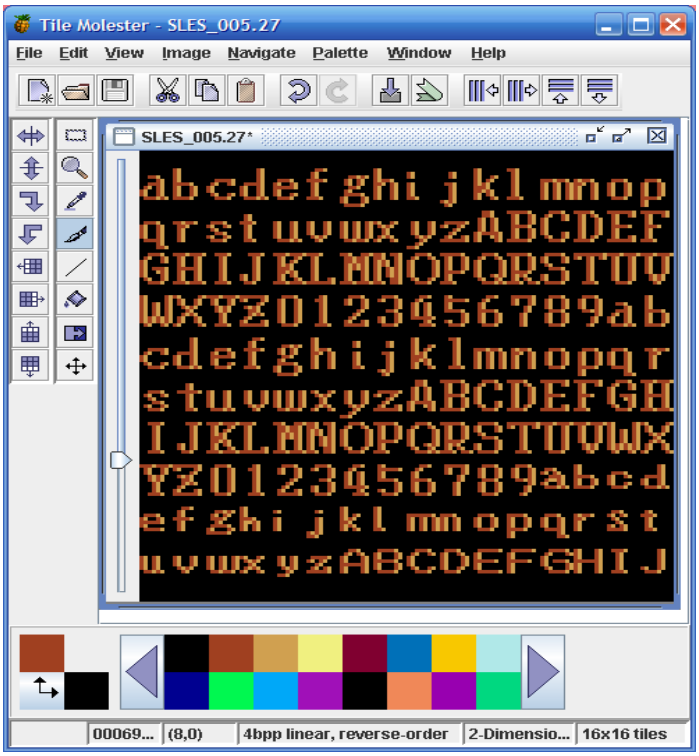
8.5. Procura e edição de gráficos

A localização e edição de gráfico em jogos de PSX é muito mais complicada que em jogos de outros consoles como o SNES e ou o Mega Drive. Para não falar que em caso de dados comprimidos sua localização pode implicar em esforços monstruosos por parte do RomHacker. Os jogos de PSX podem ter gráficos em formato RAW, ou também em TIM, ainda que seja freqüente que o modo do Game Boy do Tile Layer Pro funcione em remakes como o do Final Fantasy VI. Adiante começaremos a estudar como trabalhar com estes formatos.

8.5.1. Gráficos em formato RAW

Pelo que podemos comprovar, estes gráficos se encontram em todos os jogos 2D como os dois primeiros Suikoden ou o Breath of Fire III. A parte difícil, como com os textos, é localizar os arquivos onde estão os gráficos. Um dado a destacar é que nos arquivos MAIN.EXE e SLES_XXX que podem estar no diretório principal do CD sempre há dados bastante importantes. Assim, nosso primeiro passo será comprovar se a fonte de Suikoden se encontra em um destes arquivos. Para isso, nos valeremos novamente do Tile Molester. Tenho que dizer que foi Vegetal Gibber quem me mostro o modo de visualizar uma fonte de PSX com este programa. Vamos provar primeiro com o arquivo SLES_005.27, no qual deve estar ao fonte.

Assim, excutamos o Tile Molester como dissemos antes. Só falta saber qual codec temos que utilizar: o *4bpp linear*, *reverse-order*. Porém, lembrando que dependendo do jogo pode ser melhor optar por outro tipo de codec. Se dermos uma olhada rápida no arquivo e não encontrarmos nada, isto acontece porque devemos visualizar os gráficos no formato RAW em modo 2D, tal como foi feito na hora de modificar a fonte de Final Fantasy V de SNES. Por tanto, antes de dar uma segunda olhada, devemos clicar em *View* > *Mode* > *2-Dimensional*. Deste modo, se avançarmos pela ROM até chegar um momento em que poderemos distinguir a fonte de Suikoden sem nenhum problema. Uma vez modificada basta inserir novamente o arquivo SLES_005.27 na ISO da maneira que vimos antes para poder apreciar as trocas.

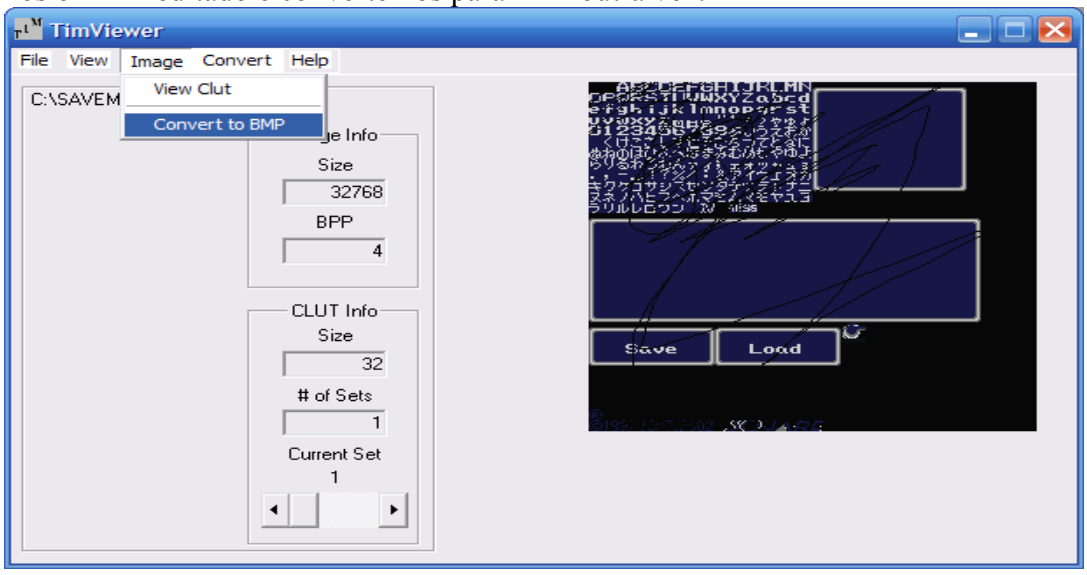


8.5.2. Gráficos em formato TIM

Para modificar gráficos em formato TIM teremos que utilizar a valiosa ferramenta TIM Viewer, que não só é capaz de visualizar imagens TIM e escanear arquivos normais em busca de imagens TIM, e ainda mais permite exporta-la para BMP e de BMP para TIM. Para ver como é fácil editar imagens em formato TIM, tomaremos como exemplo a versão de PSX do Final Fantasy IV (PAL).

Executamos o TIM Viewer e abrimos as imagens TIM com as que queremos trabalhar. É possível que se encontrem ocultas dentro de outros arquivos, porque a opção *Scan RAW File* mos será mais útil em muitas ocasiões, ainda que não seja o caso.

Em Final Fantasy IV de PSX as TIMs que nos interessa estão na pasta CARDTIM. Vamos abrir no TIM Viewer o arquivo SAVEMOD.TIM, que é no qual se encontra a fonte do jogo entre outras coisas. A primeira coisa que temos a fazer será converter a imagem TIM para o formato BMP, clicando em *Image* > *Convert to BMP*. Uma vez selecionamos a pasta destino, teremos em nosso HD a imagem que víamos no TIM Viewer mas em BMP. Agora editamos ainda que seja para destroçar literalmente a imagem a base de pixels negros por todos os lado com algum editor de imagens como o Paint. Uma vez que tenhamos salvo as alterações, vamos ao TIM Viewer, carregamos o BMP editado e convertemos para TIM outra vez:



Supondo que criamos a imagem ISO do Final Fantasy IV, inserimos o arquivo modificado com o CDMage e executamos o ePSXe. Se carregarmos a ISO modificada, poderemos ver que fizemos tudo com perfeição, já que podemos ver o seguinte:

8.6. Para saber mais

- *Extracción y utilización de CLUTs (paletas PSX) en el editor gráfico Tile Molester* (por Vegetal Gibber): Acredito que o nome já diz tudo. Se está interessado nas paletas de PSX, asseguro que este documento será de grande utilidade.
- *La FAQ de Moogle sur le hack PSX* (por Moogle). Documento que trata sobre a inserção de arquivos maiores que os originais em uma imagem ISO. No momento eu ainda não comprovei o método. Está em francês.

CAPÍTULO 9:
INTRODUÇÃO AO ASM

Este capítulo pretende apenas introduzir o leitor de maneira básica a linguagem Assembly (ASM) do processador do SNES, o 65c816, ainda que possa servir para trabalhar com outros processadores de características parecidas. Nem muito menos pretendo explicar tudo o que faz cada instrução (opcode) do processador, só explicarei algumas coisas que podem ser úteis na hora de traduzir um jogo. Já que existe muita informação disponível sobre o tema, se bem que muito técnica.

9.1. Programas necessários e outros requerimentos

- *Geiger's Snes9x Debugger*: Emulador e depurador de SNES com capacidade de escrever em um arquivo log todas as instruções que são processadas durante o tempo que especificarmos. Além disso, mostra valores de cada registro do processador de SNES durante todo o tempo.
- *Lunar Address*: Nos permitirá converter uma posição do SNES para PC (a da ROM) e vice-versa, ainda veremos seu uso neste manual.
- *SNES Professional ASM Development Kit*: Imprescindível para realizar pequenas modificações ASM dentro da ROM de SNES.
- *O jogo com que se vai trabalhar*. Neste caso tomarei como exemplo a Rom traduzida por J2e do Final Fantasy IV.

9.2. Instruções básicas

- *LDA (LoaD Accumulator), LDX (LoaD X register) y LDY (LoaD Y register)*

  Podemos dizer que o acumulador é uma variável que serve para guardar um valor que será utilizado posteriormente. Esse valor se especifica justamente depois da instrução, por exemplo, LDA #$20 carregará o valor hexadecimal 20 (#$ indica que o código esta em hexadecimal) em A. O mesmo ocorre nos registradores X e Y, ainda que estes servem normalmente como contadores para fazer bucles (logo veremos o que são).
- *STA (STore Accumulator), STX (STore X register) y STY (STore Y register)*

  Como talvez possa ser adivinhado, estas instruções se encarregam de guardar os valores do registro especificado (A, X ou Y) na direção especificado). Assim, STX $000A guardaria os valor de X na posição 000A.

- *JMP (Jump)*

  Esta é uma instrução mais que útil já que por meio dela podemos escrever nossas rotinas em outro local da ROM.

9.3. Procurar e modificar textos que não se encontram do modo normal

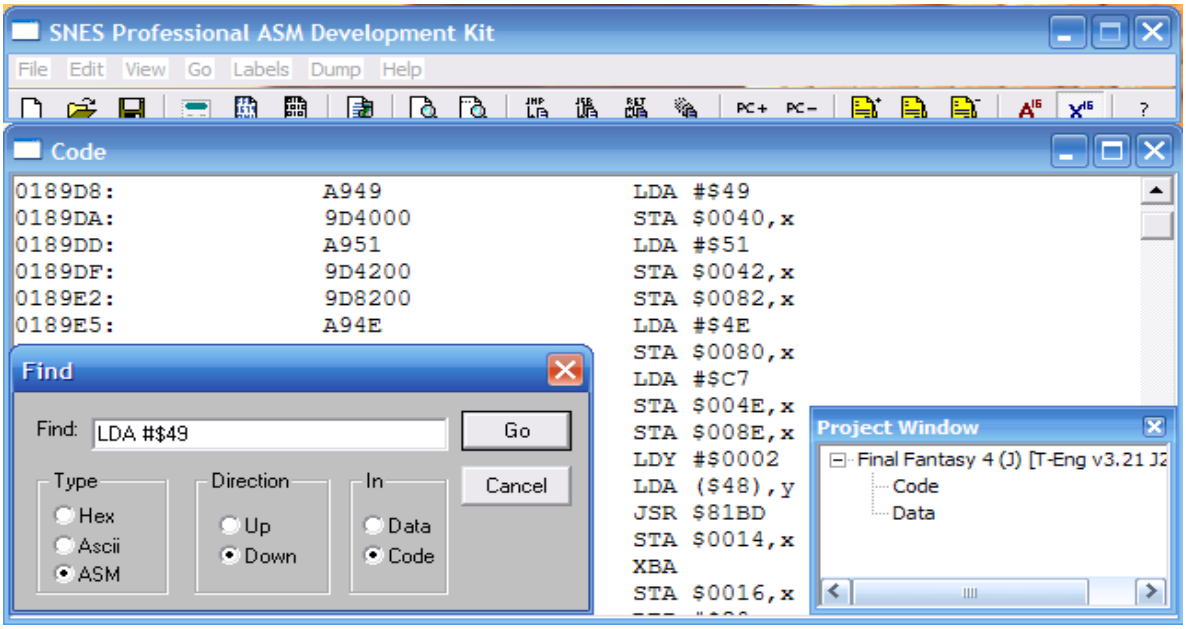
Com somente estas três instruções já somos capazes de fazer algo muito útil: procurar textos curtos que normalmente não encontram-se por métodos normais. Por exemplo, os típicos HP e MP dos RPGs que nunca são encontrados, somente são encontrados com este modo, ao menos no jogos da Square como os Final Fantasy ou o Chrono Trigger. Vamos tomar como exemplo a ROM de Final Fantasy IV japonesa já com o patch da tradução do grupo J2e aplicada.

Primeiro vamos procurar os HP e MP do menu principal para modificá-los por PV e PM respectivamente. Para isso, abrimos o SNES Professional ASM Development Kit, e clicamos em *File > New Project* e abrimos a ROM do Final Fantasy IV. Será aberto uma janela do projeto com o nome da ROM aberta. Se dermos um duplo clique no nome aparecerão dois novos elementos, *Code*, que mostra o código ASM do jogo, e *Data*, que mostra os conteúdo da ROM como um editor hexadecimal. A janela que nos interessa é a do Code.

Se dermos uma olhada na tabela do Final Fantasy IV veremos que 49=H, 4E=M e 51=P. Bom, é provável que o código que carrega HP e MP seja o seguinte:

```
LDA #$49      ; H
STA $xxxx
LDA #$51      ; P
STA $xxxx
LDA #$4E      ; M
STA $xxxx
LDA #$51      ; P
STA $xxxx
```

Claro, LDA poderia ser LDX ou LDY, ao igual que STA poderia ser STX ou STY. Bem, o que devemos fazer agora é procurar o LDA #$49, para isso clicamos em *Go > Find*. Veremos que se encontra rapidamente o seguinte:



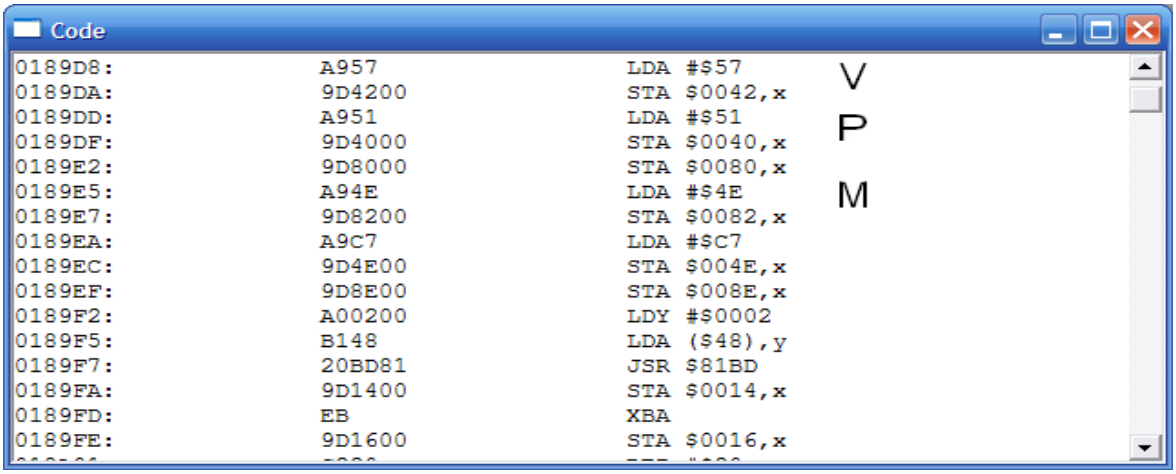
Ainda que não é exatamente a rotina que tínhamos escrito, tudo indica que estamos diante a rotina que procurávamos. De fato, está "optimizada" devido ao fato de que em vez de carregar o caractere 'P' (51=P) duas vezes, ele carrega apenas uma vez mas o guarda em dois locais diferentes. Por sorte isso não nos impõe um impedimento, já que em PV (lembre-se que 57=V) e PM o 'P' também se repete, ainda que em locais diferentes. Segundo as posições que aparecem, podemos deduzir em quais posilções são guardados os valores de HP e MP.

$0040 $0042 $0080 $0082
H P M P

Deste modo, utilizando a lógica devemos trocar os valores que são carregados e as direções onde são salvas no mesmo espaço que temos. Uma solução possível é a seguinte:

$0040 $0042 $0080 $0082
P V P M

Assim, vamos modificar o código da seguinte maneira (para editar código basta dar um clique sobre o menu e apertar a tecla INTRO quando for fazer a modificações desejadas):



Uma vez feitas as modificações pertinentes, salvamos o arquivo. Agora só resta comprovar se o que fizemos serviu para algo. Se executarmos o jogo no emulador e abrirmos o menu acabaremos com as dúvidas:

De fato, as modificações que foram feitas surtiram efeito. Devemos nos sentir orgulhosos, pois fizemos nossa primeira modificação ASM com alguns conhecimentos mais que básicos. Quem disse que deveria temer o ASM? O próximo que vamos modificar é o LEVEL que aparece no menu e que tão pouco se encontra por meios normais. Como veremos, traduzir LEVEL por NÍVEL, pode estabelecer um problema ao não haver espaço suficiente na rotina original.

Sabendo desta vez que a rotina que procuramos está “optimizada” da maneira que vimos antes, e que 4D=L, 46=E e 57=V, devemos esperar uma rotina como a seguinte:

```
LDA #$4D     ; L
STA $xxxx,x
STA $xxxx,x
LDA #$46     ; E
STA $xxxx,x
STA $xxxx,x
LDA #$57     ; V
STA $xxxx,x
```

Se procurarmos LDA #$4D encontraremos a dita rotina. Fazemos o mesmo que antes, deduzimos em que posição serão armazenados os valores de LEVEL:

```
$000A $000C $000E $0010 $0012
  L       E       V       E       L
```

Por tanto, os valores que devem ser carregados e as posições nas quais devem ser armazenados no caso de NÍVEL são as seguintes:

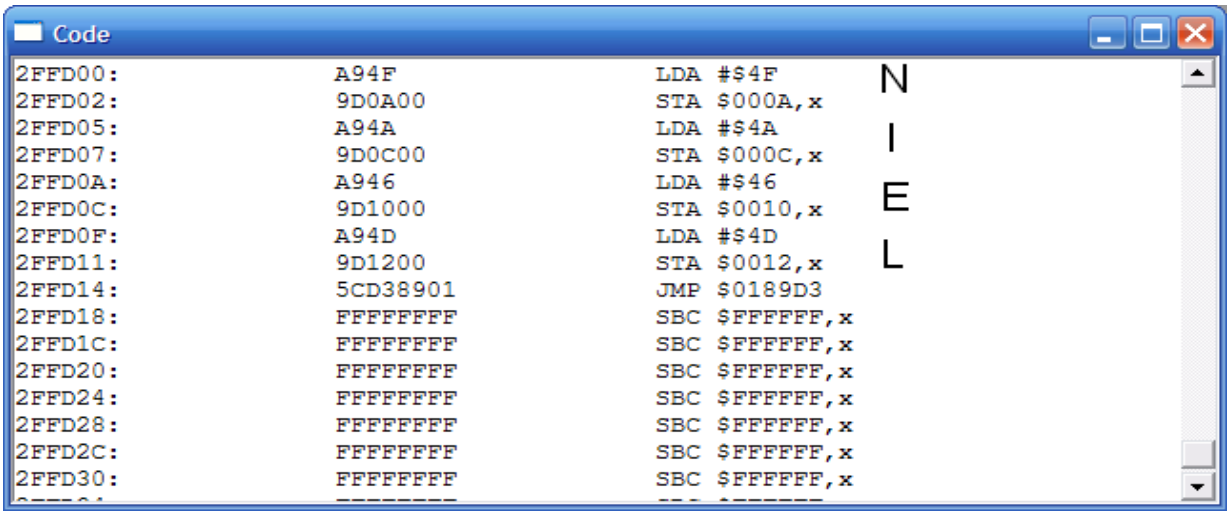
```
$000A $000C $000E $0010 $0012
  N       Í       V       E       L
```

Infelizmente, desta vez temos um problema, o 'L' e o 'E' são carregados somente uma vez e são armazenados em dois locais diferentes, mas em NÍVEL não existe nenhum caractere que se repita, porque não teríamos espaço suficiente para escrever nossa rotina. A solução para tal impedimento é procurar um local na ROM onde não exista nada, escrever ali nossa rotina e dirigir a rotina original para a nossa rotina.

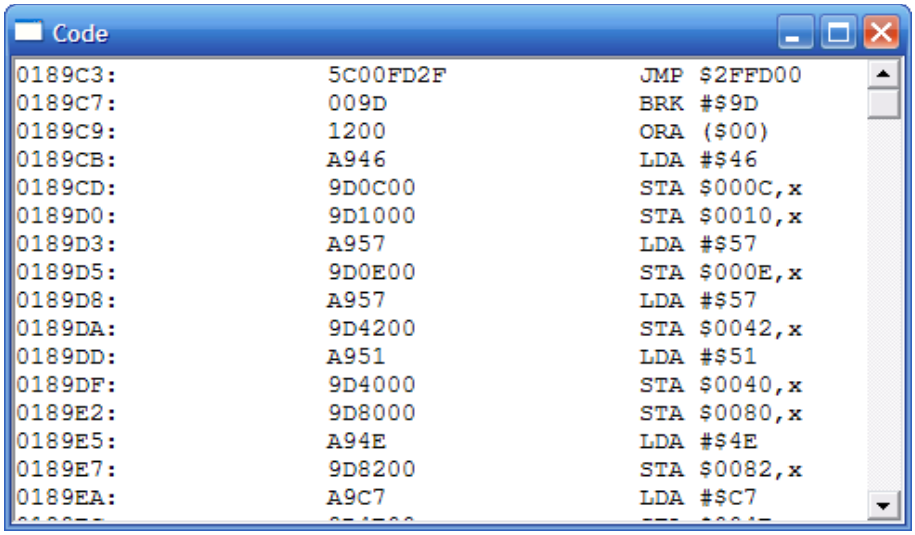
Normalmente todas as ROM possuem áreas cheias de 00 ou FF, ou seja, que estão vazias. Para encontrar uma área vazia, bastar procurar uma série de 00 ou FF na janela de Data. Não obstante, é também bastante freqüente encontrar um pouco de espaço inútil no final da ROM. Por exemplo, na posição 2FFD00 há um bloco de espaço que nos será útil.

O que devemos fazer é escrever uma rotina que carregue os valores “NÍVEL” (lembre-se que 4F=N e 4A=I) e os armazene em suas posições correspondente para que depois de finalizar a rotina vá diretamente à posição em que se carrega o caractere 'V' (0189D3), já que tanto em “LEVEL” como em “NÍVEL” estão na mesma posição e além disso é a última instrução que tem que ver com carregar “LEVEL”.

Para isto, vamos para a posição 2FFD00 na janela de Código e escrevemos nossa rotina da seguinte forma:



No final da rotina temos que colocar um JMP para a posição que carrega o caractere 'V' para que o jogo siga executando o código sem problemas. Já que só nos resta ir na rotina original e fazer um JMP para a posição da nossa nova rotina:



Como vemos com a ilustração, a instrução JMP ocupa mais que a instrução LDA, parte do código que tem continuando é trocado. Isso em principio não deve importar, pois não deve haver nenhuma rotina que salte o código como tínhamos trocado. Depois destas modificações ASM, só falta salvar as modificações e comprovar que tudo saiu bem:

9.4. Instruções avançadas: bucles

Um bucle faz referência a um conjunto de instruções que se executem X vezes segundo uma ou várias condições. Por exemplo, um detonador pode contar de 10 a 0 para fazer explodir uma bomba. Cada segundo que passa o detonador testa se já chegou ao número 0; se não, não faz nada, mas se chegar em 0, executa a ação e deve explodir a bomba. Adiante explicarei os opcodes que veremos normalmente em um bucle, ainda que exista mais:

- *INC (INCrement accumulator), INX (INcrement X register) e INY (INcrement X register)*
  Incrementa o valor da variável em questão (Acumulador, X e Y respectivamente) em um. É bastante útil para fazer um bucle se repita um número de vezes desejado.
- *DEC (DECrease accumulator), DEX (DEcrease X register) e DEY (DEcrease X register)*
  Diminui o valor da variável em questão (Acumulador, X e Y respectivamente) em um.
- *ASL (Arithmetic Shift Left)*
  Sem entrar em nenhuma explicação técnica, digamos que o que ele faz é multiplicar por dois o valor de acumulador.

- *LSR (Logical Shift Right)*

  Este opcode faz o contrário do anterior, ou seja, divide por dois o valor do acumulador.

- *CMP (CoMPare accumulator), CPX (ComPare X register) e CPY (ComPare Y register)*

  Compara o valor da variável em questão com o valor especificado. Por exemplo, CPX #$4B compara X com o valor 4B. Esta comparação será muito importante para as instruções que varemos adiante.

- *BCC (Branch if Carry Clear)*, também conhecido como *BLT (Branch if Less Than)*

  Se o valor da variável que se compra é **menor** que o valor especificado na comparação saltará para a posição especificada. Por exemplo:

  ```
  CMP #$20        ; A = #$16
  BCC $7800       ; Como o valor de A é menor que #$20, saltará a posição $7800
  ```

- *BCS (Branch if Carry Set)*, também conhecido como *BGE (Branch if Greater than or Equal)*

  Se o valor da variável que se compara é **maior ou igual** ao valor especificado na comparação, saltará para a posição especificada. Por exemplo:

  ```
  CMP #$20        ; A= #$23
  BCS $7800       ; Como o valor de A é maior que #$20 será saltada a posição $7800
  ```

- *BEQ (Branch if Equal)*

  Se o valor da variável que se compara é **igual** ao valor especificado na comparação saltará para a posição especificada. Por exemplo:

  ```
  CMP #$20        ; A= #$24
  BCS $7800       ; Como o valor de A não é igual a #$20, não será saltada a posição
                  ;  indicada e se executa a  próxima posição.
  ```

- *BNE (Branch if Not Equal)*

  Se o valor da variável que se compara **não é igual** ao valor especificado na comparação saltará para a posição especificada. Por exemplo:

  ```
  CMP #$20        ; A= #$08
  BCS $7800       ; Como o valor de A não é igual a #$20, será saltada a posição $7800
  ```

- *BRA (Branch Always)*

  Independentes de que se faça uma comparação ou não, sem importar o resultado destas, sempre saltará para a posição especificada.

- *JSR e JSL (Jump to SubRoutine)*

  Sua função é parecida com a de JMP. Salta a posição especificada e executa o código até que se encontre o opcode que faz regressar a posição seguinte ao JSR ou JSL correspondente. Bastante útil quando uma mesma rotina for utilizada várias vezes. A diferença entre JSR e JSL é que JSL é utilizado para especificar posições mais longínqüas, pois ocupa mais bytes e demora mais tempo para ser executado.
- *RTS e RTL (Return To Subroutine)*

  É o opcode que faz terminar uma rotina. A diferença entre RTS e RTL é a mesma que entre JSR e JSL e devem ser usados adequadamente.

9.5. Rastrear códigos: outras modificações ASM

Depois dessa enxurrada de novas instruções dá a impressão de que nos encontramos em ponto já bastante avançado, algo que não passa de uma ilusão para nós. Porém, como já vimos anteriormente, com alguns conhecimentos básicos se pode conseguir fazer algo. Desta vez vamos ver como modificar algo "sério" como é o que proponho para continuar. O segredo sempre está em usar o sentido comum, o ridículo e clássico método de tentativa e erro e, como não, o aprendido até o momento.

Resulta que no Final Fantasy IV, dependendo se estamos em uma cidade ou no mapa-múndi ou em uma masmorra, temos a possibilidade de salvar ou não a partida. Assim, "Save" aparecerá em cinza em uma masmorra enquanto que aparecerá em branco no mapa-múndi. O problema aparecerá quando nomearmos "Guardar" em lugar de "Save" (não há nenhum problema pois já há espaço suficiente; temos que procurar com a tabela dos diálogos, não como a do menu), pois só aparecem em cinza o quatro primeiros caracteres como podemos ver na imagem:

Tudo parece indicar que existe uma rotina que, conforme esteja na masmorra ou não, faz que apareça em cinza quatro caracteres. Com nosso conhecimento parece que embarcar na aventura de

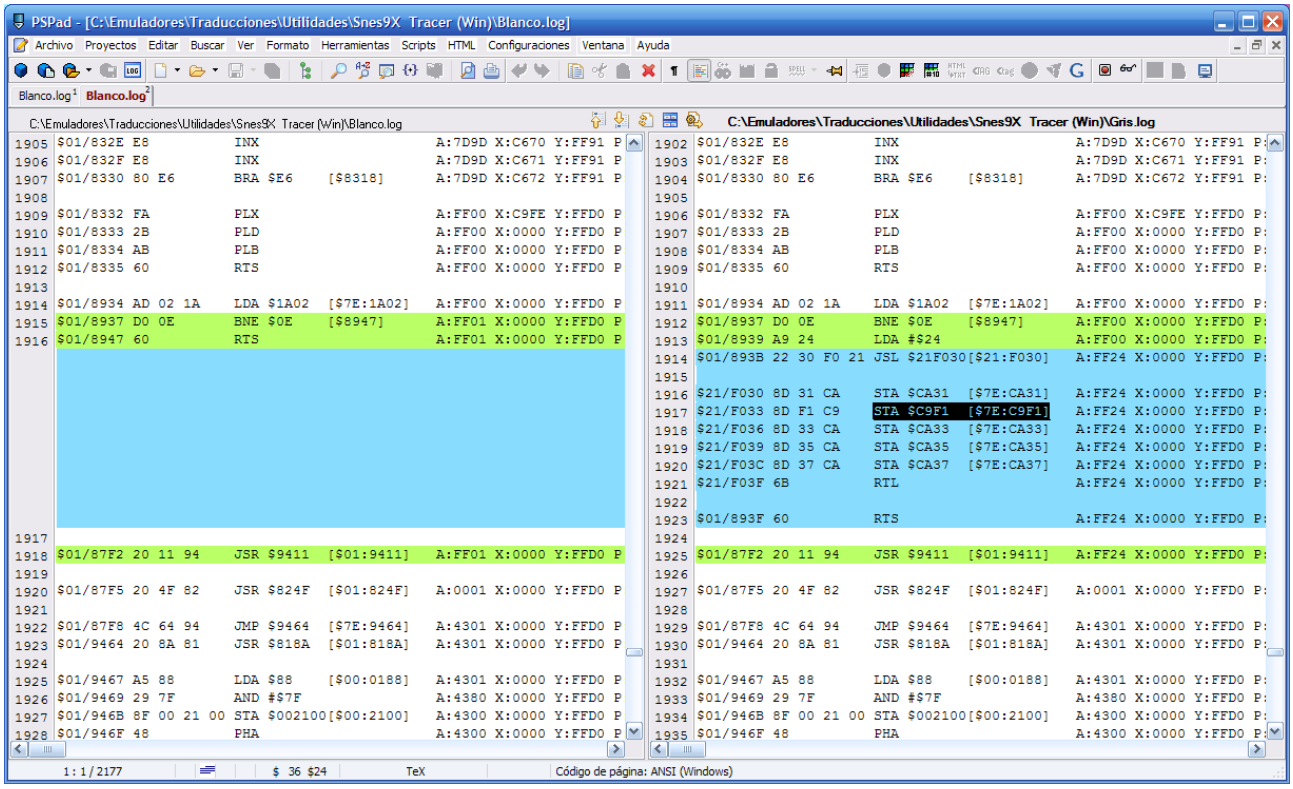
procurar e além disso modificá-la pode resultar em um tarefa impossível, mas logo comprovaremos que não é tão difícil, se bem que também não é fácil.

O que temos que fazer é criar dois logs com todas as instruções executadas no momento de abrir o menu. O primeiro log será criado em um lugar onde se pode salvar e o segundo em um lugar onde seja impossível salvar. Para isso, devemos fazer uso do Geiger's Snes9x Debugger. A verdade é que prefiro usar a versão de LordTech do Snes9x mas como é uma versão antiga é possível que não emule alguns jogos. Bem, o primeiro coisa que temos que fazer é desativar o som clicando em *Sound* > *Playback Rate* > *No Sound* e ter ativado a janela de depuração *Trace Once*, *Squelch* e *Auto Usage Map*. Para executar o jogo, basta clicar em *Run* e trocar a janela do emulador.

Para o que temos que fazer adiante, é possível que seja preciso um pouco de prática. Consiste em clicar no botão que retire o menu principal enquanto se joga e nesse preciso instante ativar a opção CPU. Uma vez ativada, deve-se trocar a janela do emulador e desativar a opção CPU nada mais irá aparecer no menu. Se fizermos tudo certo, teremos um log de uns 149kb mais ou menos no diretório em que está ROM que carregamos. Agora basta fazer está operação em um lugar onde se pode salvar a partida e em outro que não. Lembre-se que depois de criar um log é recomendável fechar o emulador e começar de novo para que o código já executado não volte a ser escrito. Se tudo sair bem, ao abrir o arquivos log (é conveniente que chame um de Branco.log e o outro de Cinza.log segundo sua correspondência) as primeiras instruções devem ser iguais.

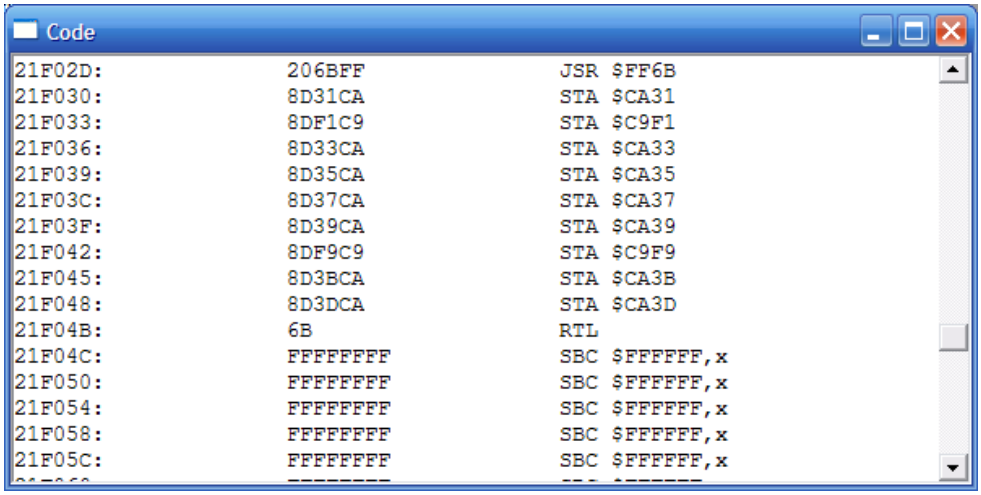
Enquanto escrevia estas linhas me dei conta que não era necessário explicar nada relativo aos bucles, salvo as instruções *JSR* e derivadas. Queria complicar o leitor o menos possível, mas o fiz, e feito está. Bom, em qualquer caso, sigamos com o que queria explicar. Abrimos o arquivos Branco.log com o programa PSPad e comparamos com o Cinza.log mediante *Herramientas* > *Diferencias en texto* > *Comparar com archivo...*. Se dermos um olhada rápida veremos que tudo coincide, menos algumas parte. Usando o senso comum nos damos conta de que a rotina que procuramos deve estar no Cinza.log porque por defeito todos os caracteres aparecem brancos. Assim, resta saber que código é o que corresponde a rotina em que estamos procuramos.

Tudo aponta que em $21/F030 está a solução. Há uma rotina que simplesmente guarda o valor do acumulador em 5 posições e além disso, ela tem haver com o que se mostra na tela nesse momento, já que $7E:xxxxsó indica isso (ou ao menos assim ocorre nos jogos que foram abordados). Mesmo sendo poucas, estas posições seguem a estrutura que vimos ao modificar LEVEL por NÍVEL.



Só que há algo que não se encaixa, e que para economizar tempo, explicarei agora. Resulta que como a primeira letra de “Save” é maiúsculas e portanto, mais alta que resto, deve colocar em cinza tanto na parte acima como na parte abaixo. Daí a explicação do STA $C9F1 [$7E:C9F1].

Bom, é hora de modificar esta rotina por uma que coloque em cinza todas as letras de “Guardar”. Basta acrescentar três instruções STA como a que vimos antes, ainda que como o 'd' de “Guardar”é também mais alta que o resto das letras será necessário acrescentar um STA extra como o que é utilizado para o 'S' de “Save”. Abrimos o SNES Professional ASM Developmente Kit, carregamos a ROM do Final Fantasy IV e procuramos a instrução STA $C9F1. Parece que estamos com sorte, pois temos espaço de sobra para inserir nossa rotina, que deve ficar como se mostra na imagem:



Espero que não haja nenhuma confusão na hora de escrever novas direções, simplesmente estão baseadas no resto. Agora já só falta comprovar se tanto esforço valeu a pena. A prova final consiste em salvar as alterações e executar a ROM modificada no emulador. Prova superada?

9.6. Para saber mais

Apesar haver muitíssima informação disponível na internet sobre ASM, prefiro não citar documentos avançados sobre tal, como aqueles que tratam da inserção de rotinas de fontes variáveis ou DTE. Só irei citar aqueles documentos que considero "essências" para ter uma forte base de ASM. Se quer uma recomendação, leia primeiro o log do canal #leetasm do IRC no qual o mestre LordTech explica as bases do ASM. De fato, eu me iniciei no ASM graças a esse log.

- *Assembly Programming for the Sega Megadrive* (por The Sega Programming Network): Incompleto, mas muito útil para conhecer as instruções básicas do Motorola 68000, o processador do Megadrive.
- *Aulas de Assembly para NES (Motorola 6502)* (por Odin): Documento em português que explica de forma amena as base de ASM do NES.
- *Curso de ASM* (por Dark-N): Dos poucos documentos em espanhol que tratam o assunto ASM. Não se pressupõem nenhum conhecimento anterior sobre o tema, o que facilitará um aprendizado de forma progressiva conforme se entra em detalhes técnicos.
- *Log del canal #leetasm* (por, entre otros, LordTech): A melhor forma de aprender sobre ASM do SNES. Ler este log será um prazer para os olhos que anseiam por conhecimento.
- *Ze Skeud's guide sur l'asm* (por Skeud): Ainda que precise de conhecimentos prévios de ASM, trata algumas rotinas tomadas de diferentes consoles que pode servir de apoio ao aprendizado de ASM.

APÊNDICE:
CRIAÇÃO E APLICAÇÃO DE PATCHS

Já que criar e aplicar patches pode acontecer em qualquer momento, decidi proceder sua explicação ao final deste manual neste apêndice. Antes de nada, devo declarar que para ROMs são utilizadas patches no formato IPS e para as imagens ISO de PSX são utilizados o formato PPF. Assim pois, começo a explicação.

1.1. Programas necessários e outros requerimentos

1.1.1. Formato IPS

- Para criar o patch: Snestool, o arquivo original e o arquivo modificado.
- Para aplicar o patch: Snestool e o arquivo original.
- Para corrigir o checksun: WindHex 32 para SNES e GenRomSuite para Megadrive.

1.1.2. Formato PPF

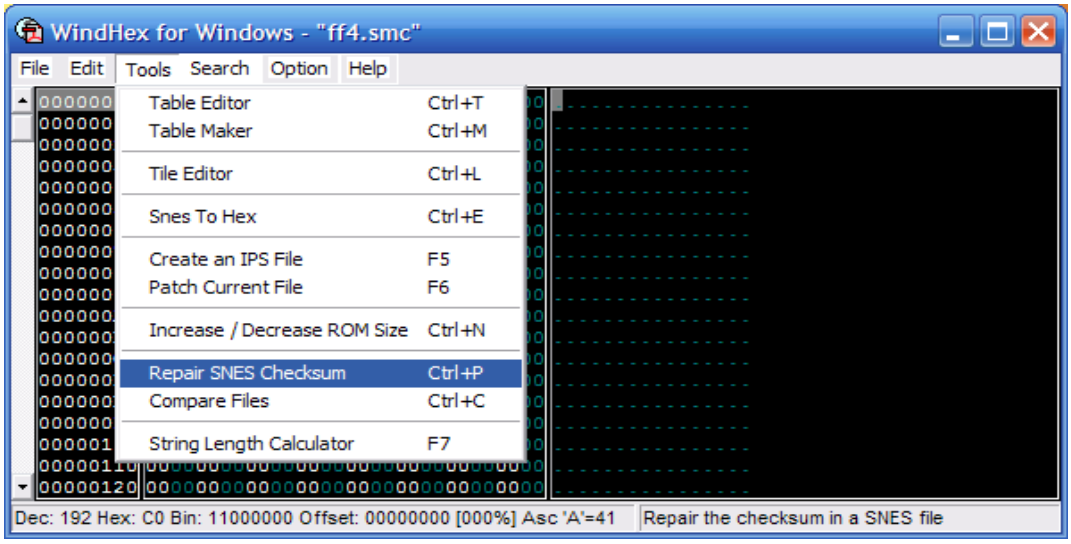
- Para criar o patch: Criar PPF, a Isso original e a ISO modificada.
- Para aplicar o patch: PPF-O-MATIC e a ISO original.

1.2. O Checksun

O "Checksun" é um valor calculado a partir de todos os bytes de um arquivo. Por tanto, se modificar o código ao traduzir, o checksun não corresponderá ao original, assim, teremos que recalculá-lo. Não é uma explicação muito técnica mas creio que cumpre seu trabalho. Os emuladores não têm problema com isso, mas os "copiadores", ou seja, aqueles aparatos que nos permitem jogar a ROM em um console real têm. Explicarei como corrigí-lo em jogos de SNES e de Megadrive; não deve ser difícil encontrar informação sobre outros consoles na Internet caso seja necessário.

1.2.1. Corrigir o checksun de jogos de SNES

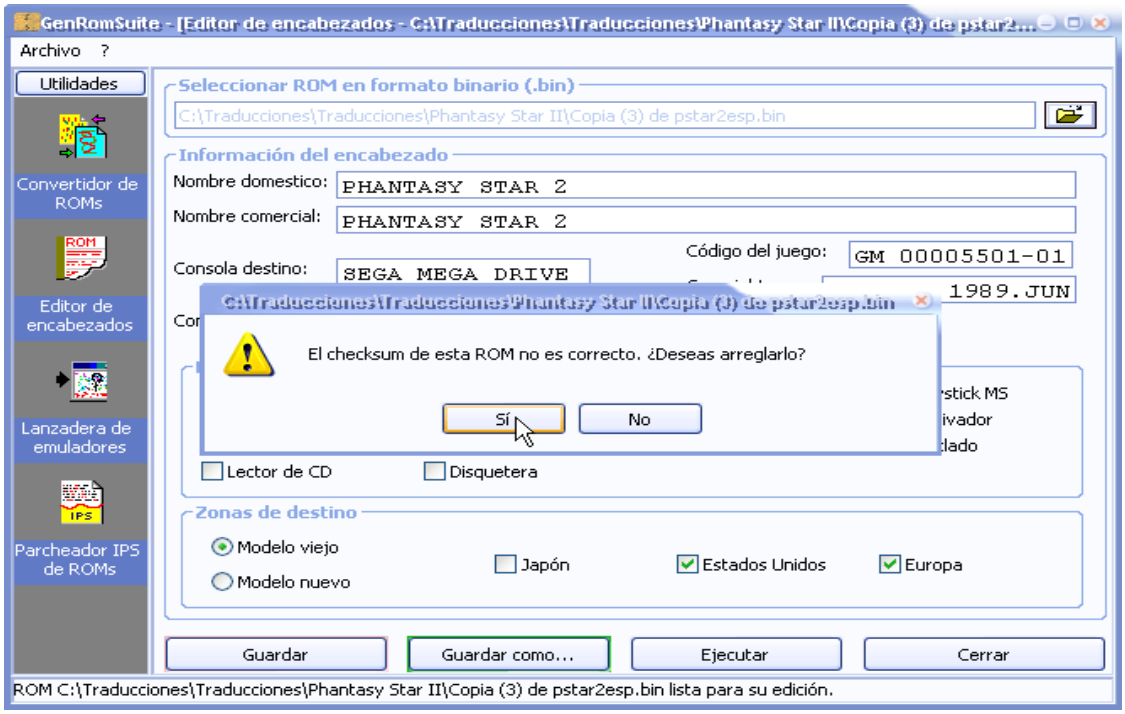
Há vários programas que permitem fazê-lo, mas eu gosto do WindHex32 por sua facilidade. A única coisa que temos que fazer é carregar a ROM e clicar em *Tools > Repair SNES Checksun* como mostra a imagem a seguir e depois salvar a imagem:



1.2.2. Corrigir o checksun de jogos de Megadrive

É provavel que nuca faça falta, porque os emuladores Kega Fusion e Gens possuem a opção de corrigir ao carregar a ROM, ainda que francamente nunca consegui corrigir o checksun com nenhum emulador, apesar de aparecer uma mensagem que informa que foi corrigido.

O que temos que fazer é ir ao menu *Editor de Encabezados* e carregar a ROM. A menos que o checksun esteja correto, aparecerá uma mensagem em que perguntará se quer corrigí-lo. Não há que preocupar-se com a mensagem de advertência que aparece depois.



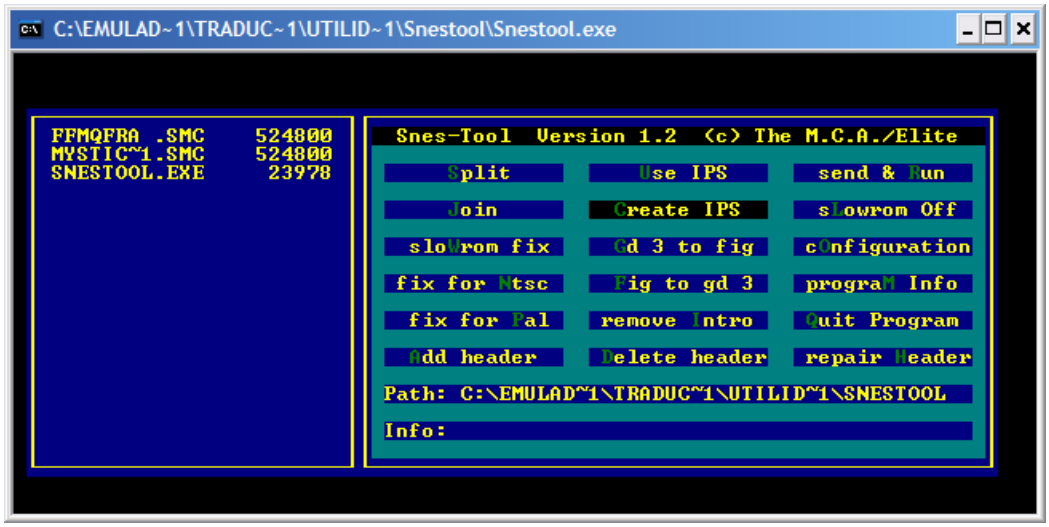
1.3. Patches IPS

1.3.1. Criar patches

Simplesmente clique em *Create IPS*, aperte a tecla ENTER, selecione a ROM original, aperte novamente ENTER e por último selecione a ROM modificada para apertar peça última vez ENTER.

1.3.2. Aplicar patches

Clique em *Use IPS*, aperte ENTER, selecione a patche, aperte ENTER e por último selecione a ROM original para apertar pela última vez o ENTER.

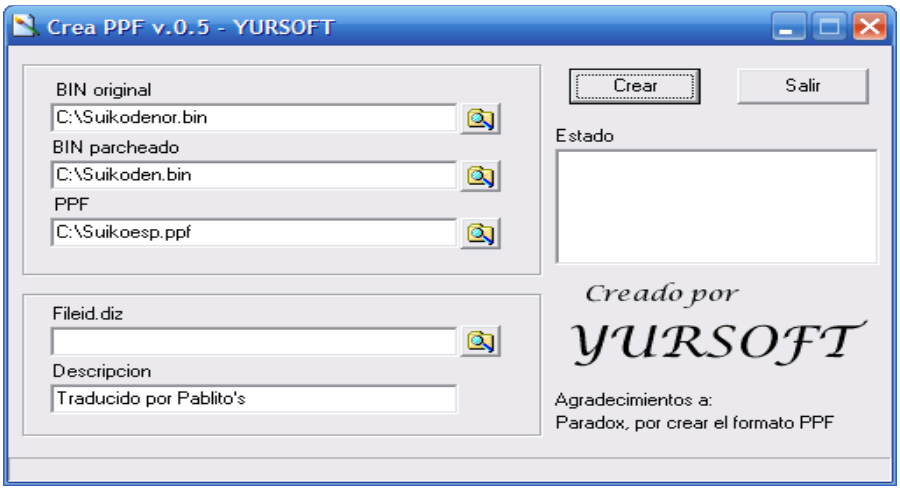


1.4. Patches PPF

Apêndice: Criação e aplicação de patchs

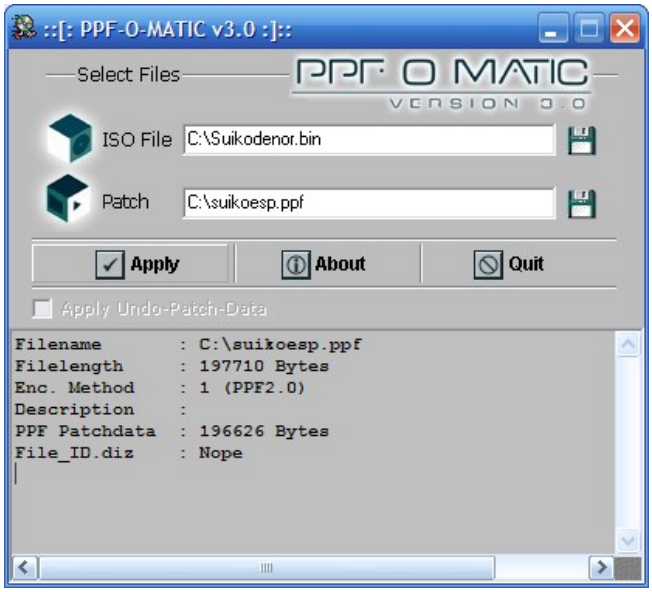
1.4.1. Criar patches

Abrimos o Crear PPF e selecionamos a Isso original, a modificada e especificamos o arquivos PPF a criar. Além disso, também podemos especificar uma descrição do patche, ainda que está limitada a 50 caracteres. Uma vez com tudo isso pronto, basta clicar em *Crear* e esperar alguns minutos já que o processo leva tempo (com um Celeron 1,73 GHz demorou de 4 a 5 minutos).



1.4.2. Aplicar Patches

Tão fácil como especificar a ISO original e o patche no programa PPF-O-MATIC e clicar em *Apply*. Se aparecer uma mensagem de erro selecione o patche, assegure-se de que tudo esta correto.



APÊNDICE II: NOTAS DA TRADUÇÃO

Notas

Esta seção não faz parte do documento original. Ela foi criada com o intuito de expor algumas alterações feitas pelo tradutor, onde o mesmo achou necessário explicar os motivos das alterações.

- Optamos por não substituir a maioria das imagens do documento, mas caso isso se faça necessário, faremos a substituição em uma próxima versão do mesmo.
- Foi alterado o link para a licença do documento, o link aponta para a mesma versão, mas para o idioma português.
- Foi alterado o link para a página do grupo Sayans para o endereço atual do grupo.
- O texto na seção 1.4.1 foi alterado radicalmente, pois no português não é usada acentuação no inicio de frases:

  Texto Original:

```
      Es inadmisible utilizar los signos de exclamación e interrogación
sólo al final:
      Ven aquí! No me oyes?        ¡Ven aquí! ¿No me oyes?
```

Créditos

Tradução para o Português (BR)

Versão 1.0

Tradução por Marciso Gonzalez (RiFF)

Edição/Revisão por Israel Crisanto (Fallen_Soul)