¿ Ahora qué?

Mi introducción al mundo de la programación fue por medio de la comunidad de "rom-hacking" de Pokémon. Siempre me llamo la atención el rom-hacking, la modificación y creación de mapas, y luego descubrí el lenguaje de scripting/programación que utilizan los juegos de tercera generación de Pokémon de la consola Game Boy Advance (GBA). Para la modificación y creación de scripts se utiliza una herramienta llamada XSE, que nos permite realizar estos scripts por medio de un lenguaje/dialecto muy similar a assembly, en el fondo estos juegos utilizan sus propias versiones de assembly , en el game boy original y el color se utilizaba una versión de z80 assembly. Al estar basado en assembly, tiene varias de sus características, como el uso de "punteros" y tener que utilizar ciertos espacios de memoria libre.

Un fragmento de codigo seria el siguiente:

```
#dynamic 0x800000

#org @main
<Contenido de nuestros scripts>

end
```

Con ""#dynamic" estamos diciendo que usaremos cualquier espacio disponible después del "offset" 0x800000 , con "#org" definimos que comenzaremos un nuevo script al que nos referiremos con el puntero @main.

Así mismo, XSE contiene otras características como el uso de " flags", registros que su valor cambia cuando ciertos "eventos" ocurren, condicionales, subrutinas , etc.

El motivo por el que se trabajaba con esta clase de lenguaje era debido a las limitaciones de hardware con las que se tenían que enfrentar los desarrolladores, teniendo entonces la necesidad de tener que comunicarse lo mas directamente posible con el hardware y no poder permitirse una capa de abstracción mayor debido a que demandaría mejores características que las sencillas consolas portátiles de ese entonces no les podían proveer .

En el momento en que me llamaba la atención el rom-hacking de Pokémon nunca logre a comprender bien varios conceptos que XSE utilizaba, como el uso de registros que se encuentran libres, flags, punteros entre otros, ya que se nos pidió realizar este ensayo fui a repasar algunos scripts que había hecho yo mismo, hace varios años y algunas guías sobre el tema y realmente al ser una versión de Assembly, es fácil darse cuenta de lo mucho que se parece a este, los conceptos y herramientas que utiliza, así como la lógica que se suele seguir cuando estamos programando en este lenguaje y logre comprender muchas de las cosas que no comprendí cuando comencé a programar.

Debido a las diferentes prácticas y ejercicios que realizamos este ciclo, me fue muy fácil poder comprender como funciona el lenguaje que estos juegos utilizan, porque se da la necesidad de tener que "reservar" memoria nosotros mismos, que registros tenemos que manipular o que banderas se ejecutan y en que momento, todos estos elementos son necesarios para poder utilizar el lenguaje de manera correcta.

Concluyendo

Aunque bien, fue divertido regresar y poder comprender como funcionaba la programación/scripting de uno de mis juegos favoritos, creo que es interesante ver como Assembly aun tiene un lugar importante en diferentes ámbitos relacionados al desarrollo de software, y es especialmente útil cuando nos encontramos con muchas limitaciones de Hardware o cuando tenemos una alta necesidad de poder comunicarnos con esta, si bien la mayoría de videojuegos de hoy en día utilizan c++, creo que Assembly nos demuestra lo importante que es el uso correcto de la memoria y la comunicación que hay con el hardware, elementos que aun son importantes en el desarrollo de videojuegos.