Un poco de contexto sobre QWeb:

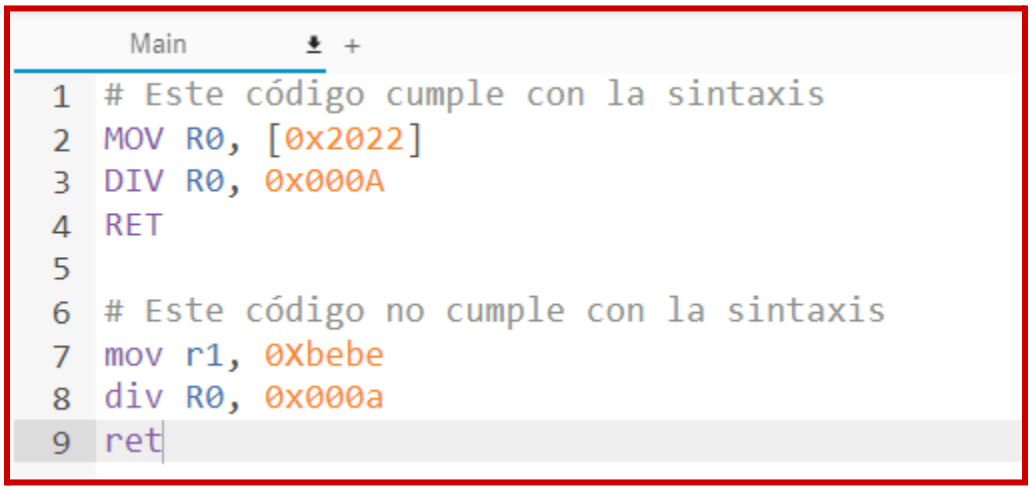
Así como la arquitectura Q fue creada para explicar el funcionamiento de las computadoras, necesitábamos una herramienta que nos permita ejecutar el código que desarrollemos. Para ello se creó QWeb, un simulador online para el lenguaje Q, desarrollado por estudiantes de la facultad.

Si quieren leer más acerca del desarrollo del simulador, tienen la documentación (técnica) del simulador en el blog. Dicha documentación está orientada al desarrollo de la misma, por lo que menciona tecnologías y decisiones de diseño, por lo que si quieren leerlo consideren que no encontrarán documentación orientada al uso de la misma.

Insensibilidad ante las mayúsculas:

Si bien el simulador ejecuta código en lenguaje Q, el código que puede ejecutar no es del todo igual al código visto en la materia, dado que permite escribir código fuente (instrucciones) con minúsculas, y ejecutar el código de igual manera.

Les mostramos un ejemplo de dos códigos equivalentes, pero donde uno cumple con la sintaxis de Q, y otro no. Les recomendamos usar la sintaxis correcta.



Comentarios del código:

Como comentamos en el archivo de recomendaciones para el TP, pueden usar comentarios dentro de su código con el fin de comunicar detalles sobre el algoritmo que están programando. Ahora bien, el simulador de Q tiene su propia

sintaxis sobre los comentarios, los cuales son comentarios de una línea que deben empezar con un **#**, por ejemplo:

```
# Esto es un comentario válido para el simulador
```

Indentación del código:

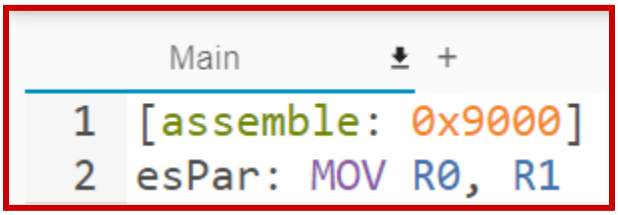
Comentamos en el archivo de recomendaciones para el TP que una buena práctica es tener una indentación definida, y usar la misma para todo el proyecto.

Sin embargo, el simulador no soporta indentaciones, por ende, todo el código que escriban debe estar sin indentaciones. Les mostramos un ejemplo del error de sintaxis al indentar:

Ensamble de rutinas en QWEB:

Luego de escribir un programa en Q, lo que se hace es ensamblarlo en memoria para su futura ejecución.

Para lograr eso en QWeb, podemos usar la anotación **[assemble: 0xHHHH]** en la línea previa a la etiqueta de la rutina que queremos ensamblar, reemplazando los HHHH por la dirección de la celda desde la que queremos ensamblar, por ejemplo, para ensamblar la rutina **esPar** (no daremos una implementación acá) desde la celda [0x9000], se hace de la siguiente manera:

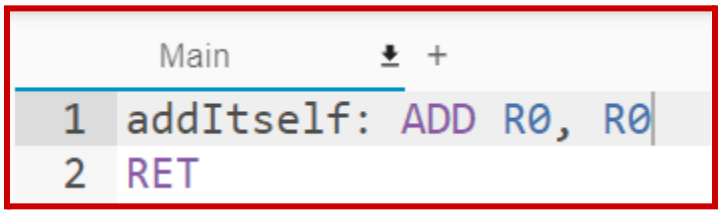


Excepción del RET respecto a la teoría:

En la teoría de la materia vimos que toda rutina tiene cierta estructura, compuesta por las siguientes partes

- Una etiqueta, para indicar el inicio de la rutina y poder identificarla
- Un cuerpo, en dónde se realiza el algoritmo diseñado para resolver la problemática
- Una instrucción RET que indica el fin de la rutina

Un pequeño ejemplo para refrescar:



Ahora bien, en el ejemplo de arriba, la rutina está implementada, sin embargo, si intentamos ejecutar la rutina, el simulador nos mostrará un error, dado que al ejecutarse la instrucción RET de la rutina, se intentará volver a la rutina desde la que fue invocada, la cuál no existe en el código mostrado anteriormente.

Para evitar este error, debemos invocar la rutina desde otra para que se ejecute correctamente, haciendo una suerte de programa y procedimiento tal cual hacíamos en EPyL, en dónde el programa principal (main) invoca los procedimientos que resuelven tareas puntuales. Les dejamos un ejemplo de cómo se vería: