Flujo de Ejecución de un programa

Organización de las Computadoras - Franco Garcino Ruiz - 2020

Flujo de ejecución en Q

El flujo de ejecución de nuestra máquina **Q** (sin importar la versión) será secuencial, es decir, se ejecutarán las instrucciones en el orden en que se lean. Este flujo podrá desviarse solo en dos casos:

a. Cuando se ejecute la instrucción *CALL* o *RET*
b. Cuando se ejecute algún salto (condicional o incondicional)

En esta presentación se mostrarán ejemplos de todos los casos

Flujo sin desvíos

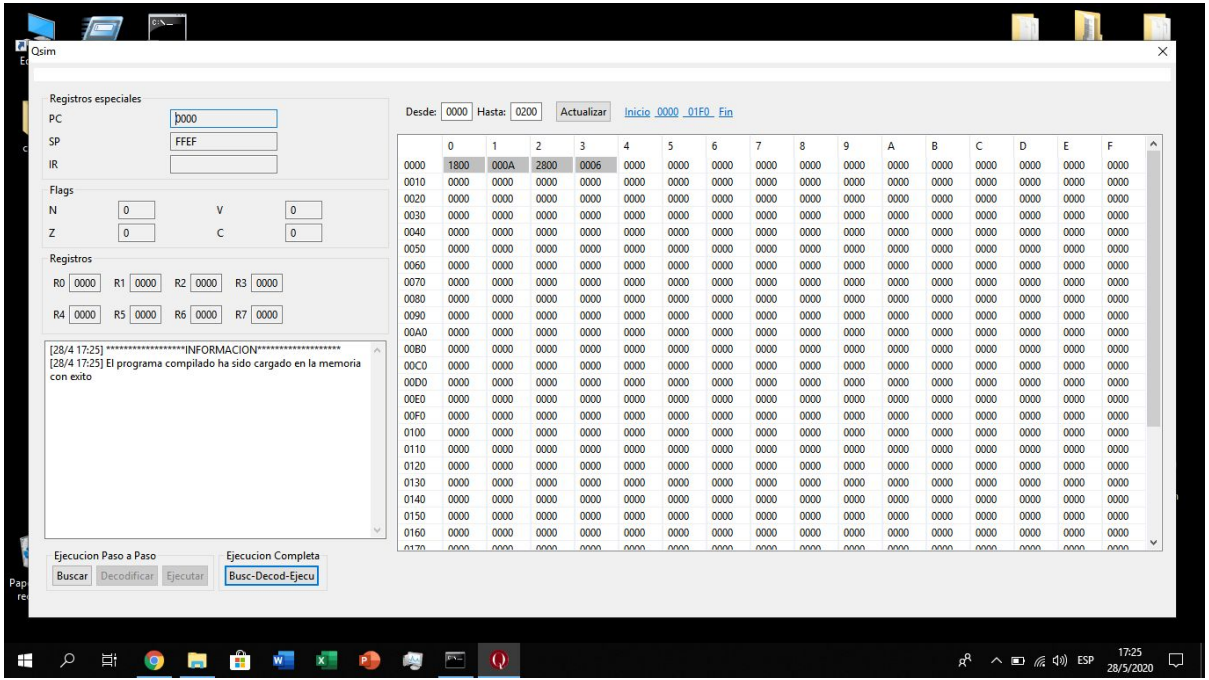
Un flujo sin desvíos en la ejecución ejecutará una instrucción tras otras en el orden que se presenten.

Ejemplo: El siguiente código está ensamblado desde 0x0000, y PC contiene la dirección 0x0000

```
MOV R0, 0x000A
ADD R0, 0x0006
RET
```

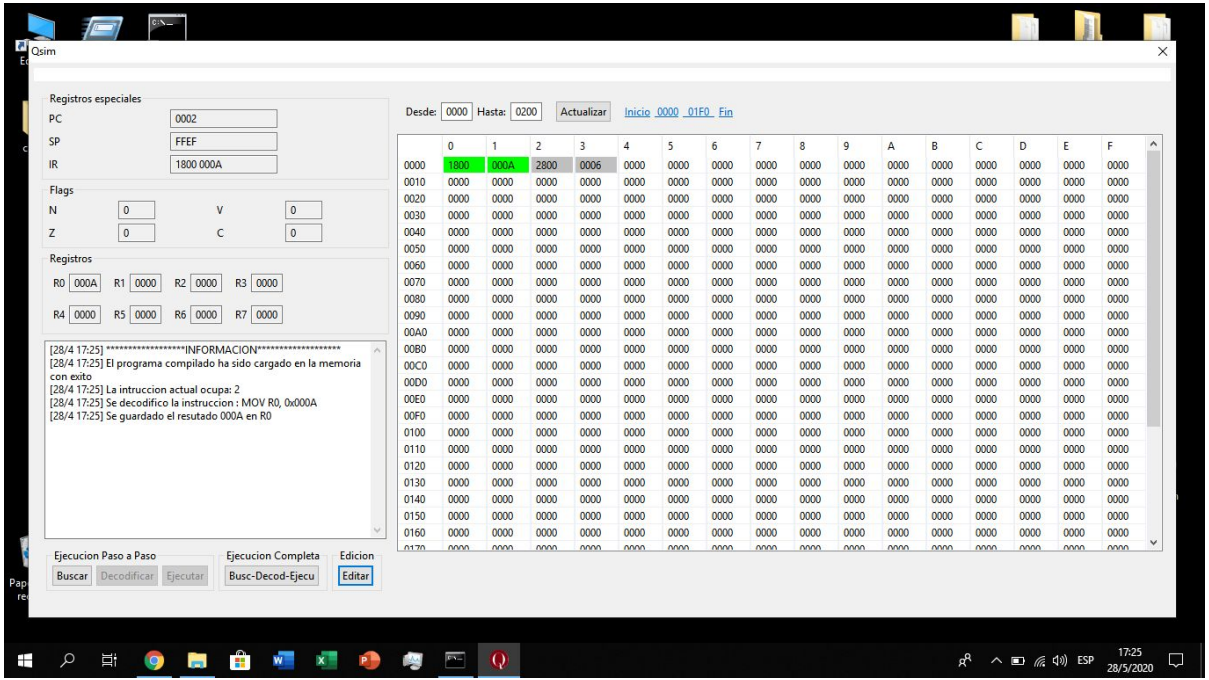
Tomando el ejemplo anterior, acá podemos ver cómo está ensamblado el código desde la celda 0x0000 en la memoria.

En la parte izquierda del simulador pueden ver los valores que poseen actualmente los registros PC, SP e IR. Además del estado de los flags, los valores almacenados en los registros [R0..R7] y una descripción de la instrucción leída.



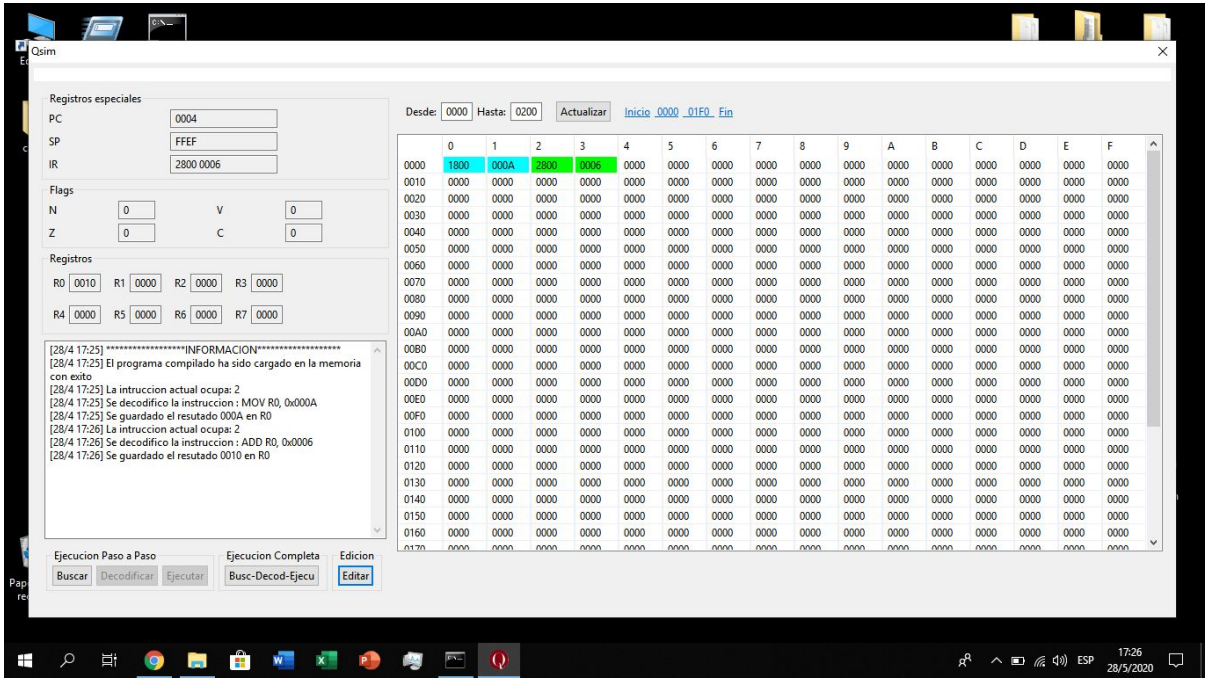
Tras la primera instrucción, vemos los siguientes cambios:

a. **PC se actualiza a 0002 (cambio secuencial)**
b. IR contiene la primer instrucción
c. R0 actualiza su contenido a 000A



Tras la siguiente instrucción, vemos los siguientes cambios:

a. **PC se actualiza a 0004 (cambio secuencial)**
b. IR contiene la segunda instrucción
c. R0 actualiza su contenido a 0010



¿Cómo es el flujo de ejecución en este ejemplo?

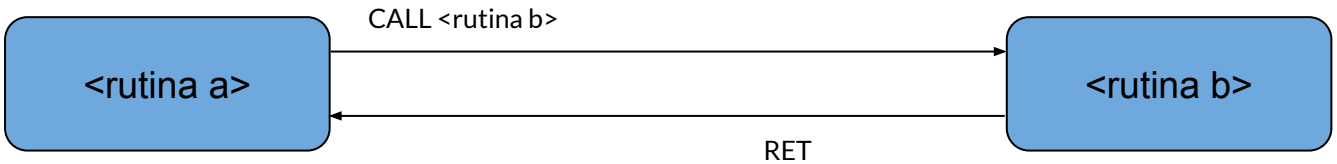
El flujo de ejecución de este programa es:

MOV R0, 0x000A	PC: 0000 → 0002
ADD R0, 0x0006	PC: 0002 → 0004

Es decir, el flujo se ejecutó de manera secuencial.

Flujo con CALL y RET

Cuando se ejecuta una instrucción CALL lo que se está haciendo es **desviar** el flujo de ejecución de la rutina principal hacia otra ensamblada en la memoria, ejecutarla, y volver para seguir ejecutando la rutina principal.



PC 2 Y PC 3

Junto con las instrucciones CALL y RET, Q3 añade entre otras cosas dos estados nuevos de PC en la simulación de un programa (Sección 7.3 del libro):

a. PC 2: es el valor que toma PC para ejecutar la siguiente instrucción en orden secuencial
b. PC 3: es el valor que toma PC tras la ejecución de la instrucción, siendo este valor la próxima celda a leer

Ejemplo

Supongamos entonces que se quiere calcular el triple de un valor, convenientemente se tiene una rutina que calcula el triple del valor almacenado en R0. Se arma entonces el siguiente código y datos:

a. PC contiene 0000
b. La rutina principal está ensamblada desde 0x0000
c. La rutina *tripleSinMul* está ensamblada desde 0x0010

Ejemplo

```
MOV R0, 0x001D
CALL tripleSinMul
RET

tripleSinMul  ADD R0, R0
              ADD R0, R0
              RET
```

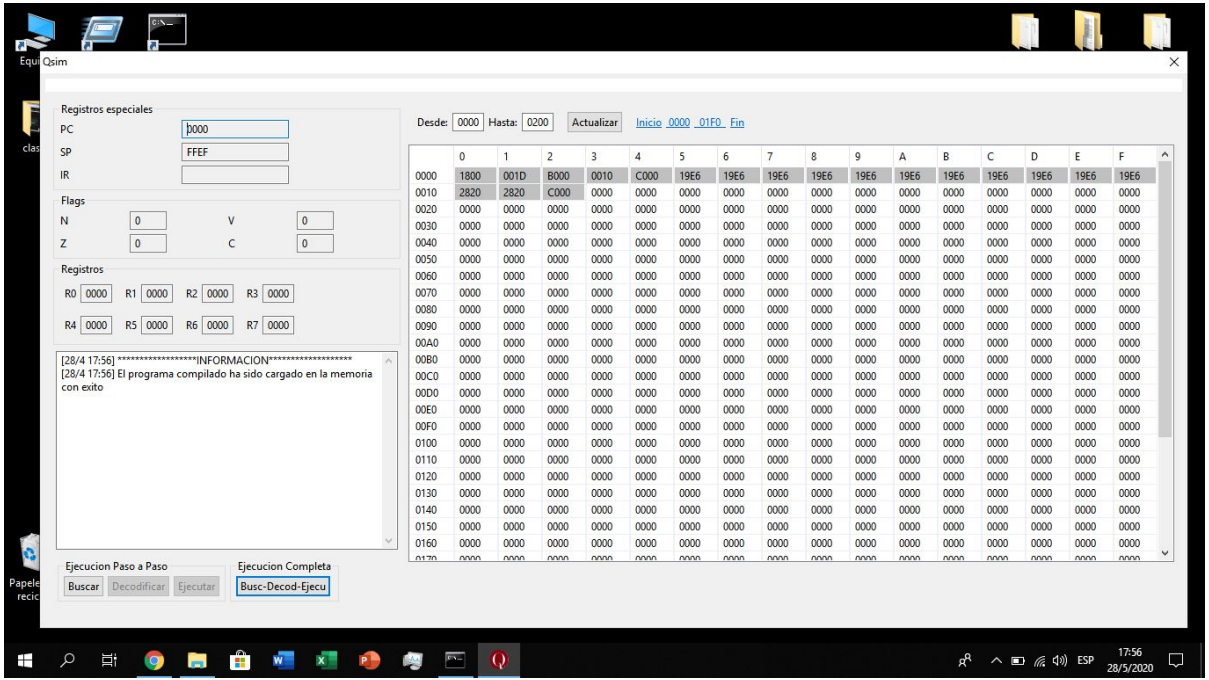
—

En esta imagen se puede observar el programa ensamblado y cargado en memoria.

Nuestra rutina principal se encuentra ensamblada desde la celda 0x0000 hasta la celda 0x0004.

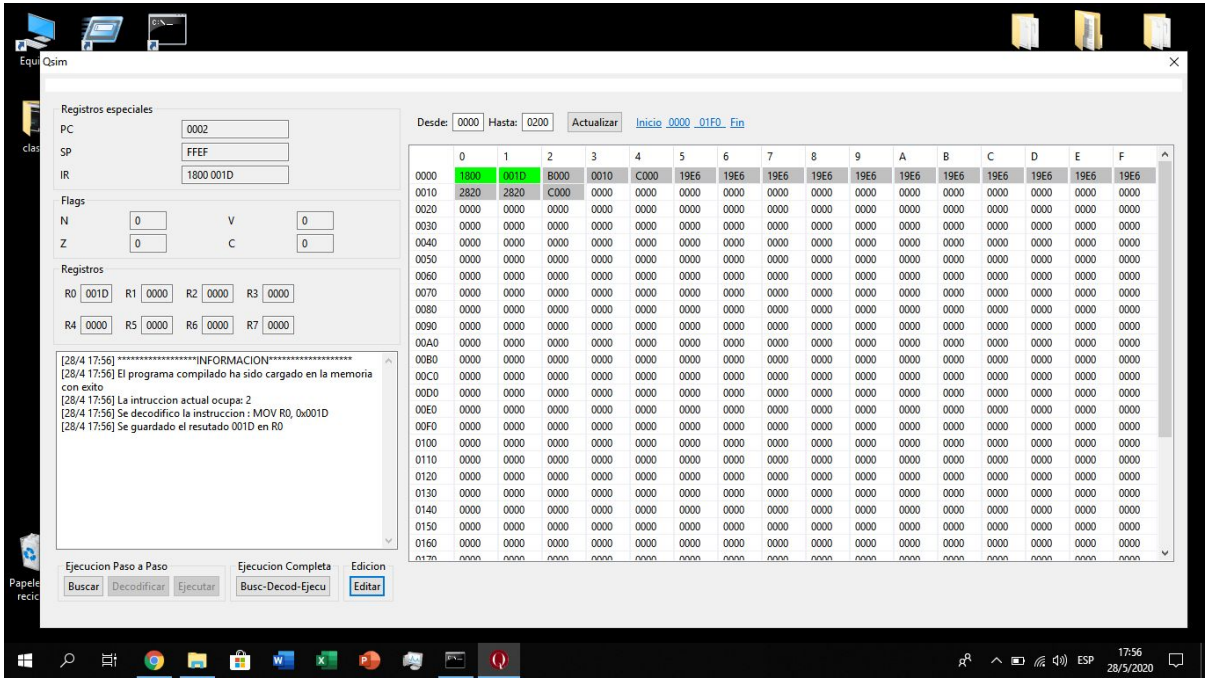
La rutina *tripleSinMul* se encuentra ensamblada desde la celda 0x0010 hasta la celda 0x0012.

PC posee el valor 0000.



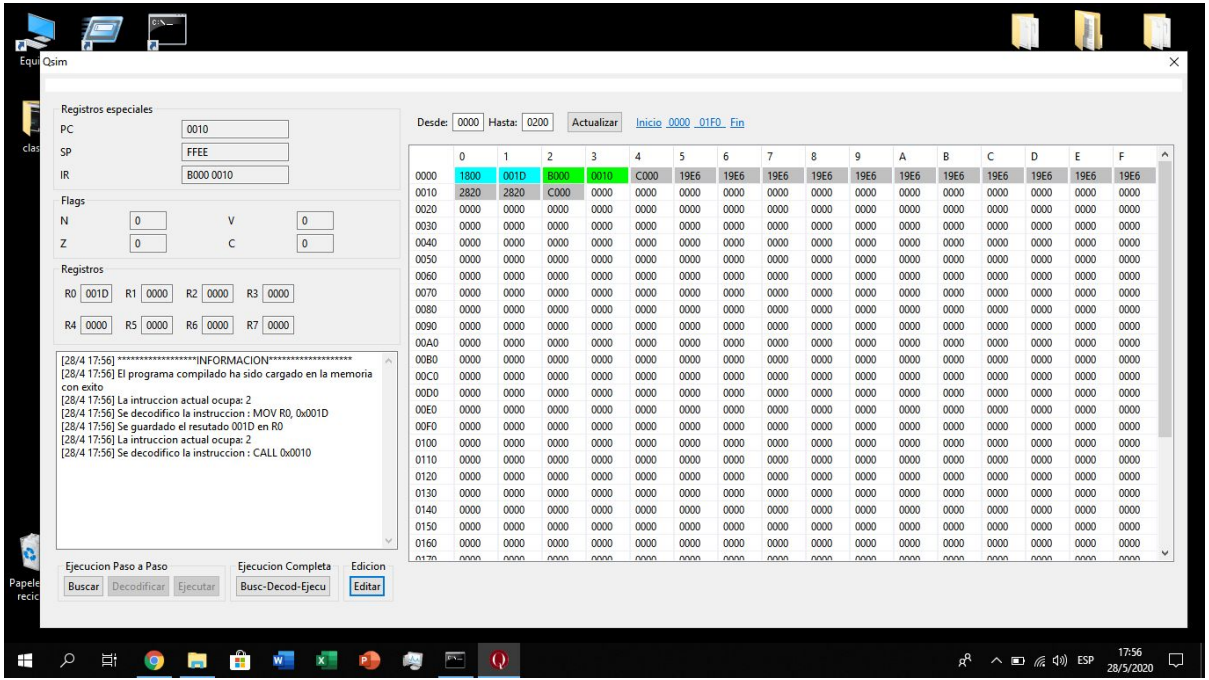
Tras la ejecución de la primer instrucción vemos los siguientes cambios:

a. **PC posee el valor 0002 (cambio secuencial)**
b. IR se actualiza su contenido
c. R0 posee el valor 001D



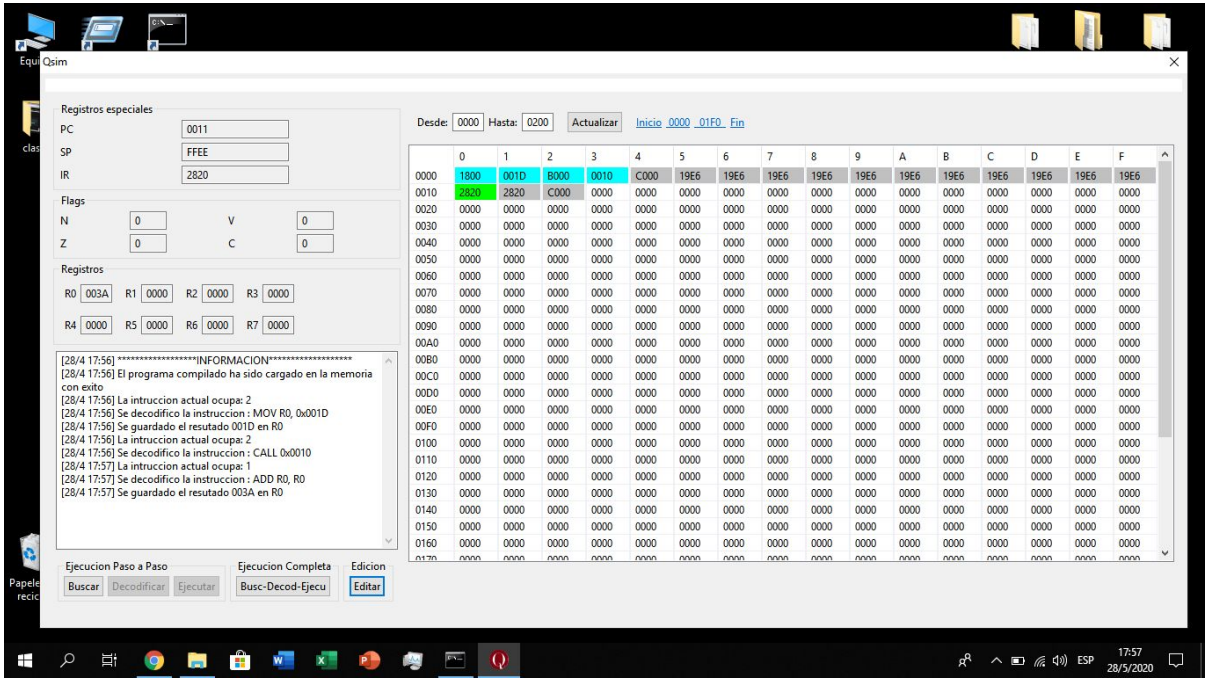
Tras la ejecución de la siguiente instrucción vemos los siguientes cambios:

a. **PC posee el valor 0010 (desvío del flujo causado por el CALL)**
b. IR actualiza su contenido
c. La pila almacena el valor 0004 (PC 2) y cambia el SP a la celda 0xFFEE



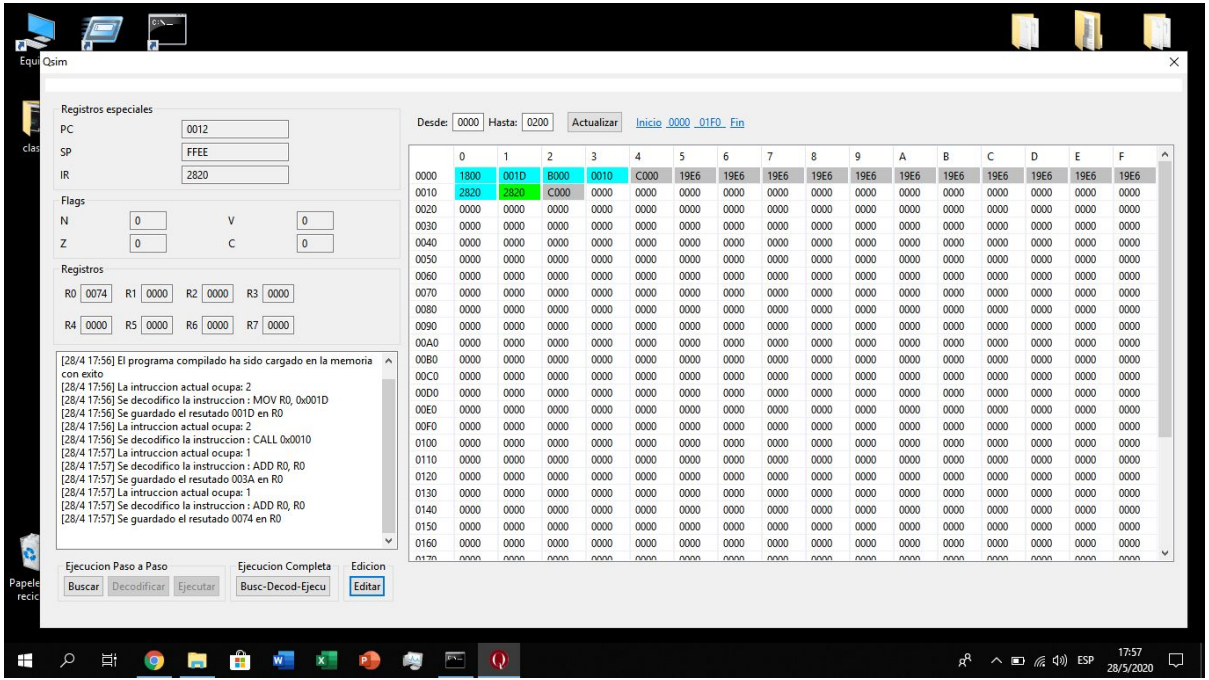
Tras la ejecución de la siguiente instrucción vemos los siguientes cambios:

a. **PC posee el valor 0011 (cambio secuencial)**
b. IR actualiza su contenido
c. R0 ahora contiene el valor



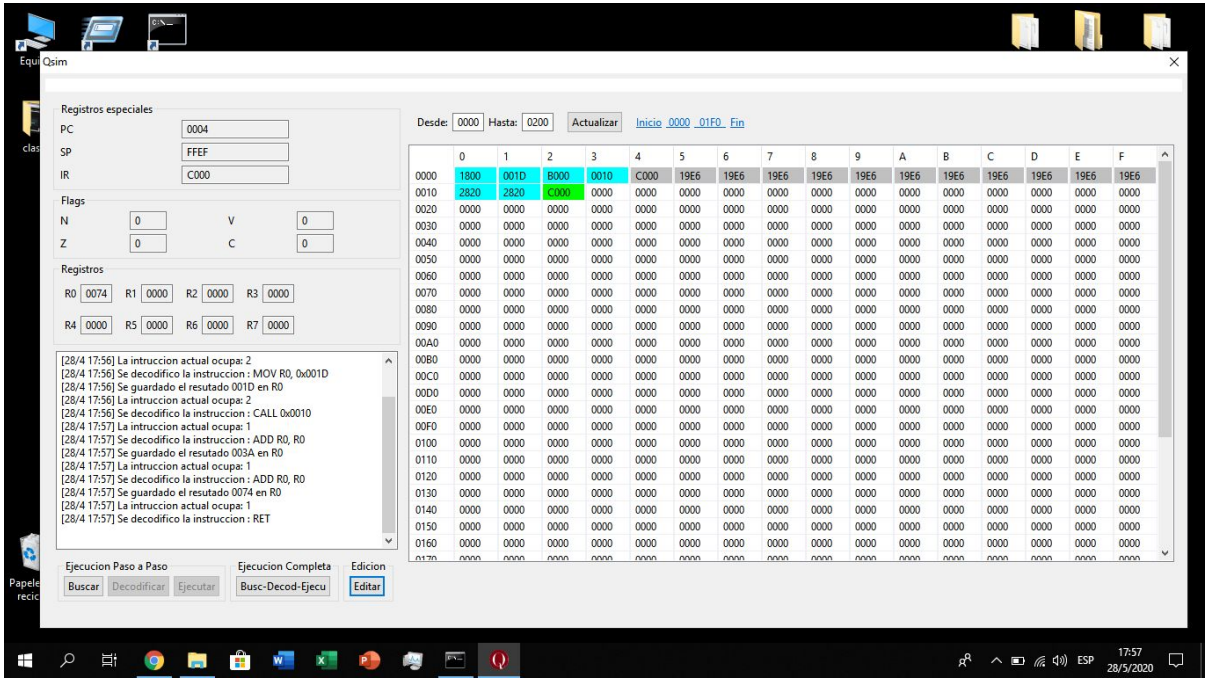
Tras la ejecución de la siguiente instrucción vemos los siguientes cambios:

a. **PC posee el valor 0012 (cambio secuencial)**
b. IR actualiza su contenido
c. R0 ahora contiene el valor 0074



Tras la ejecución de la siguiente instrucción vemos los siguientes cambios:

a. **PC posee el valor 0004 (desvío causado por la instrucción RET)**
b. IR actualiza su contenido
c. SP ahora apunta a la celda 0xFFEF



¿Cómo es el flujo de ejecución en este ejemplo?

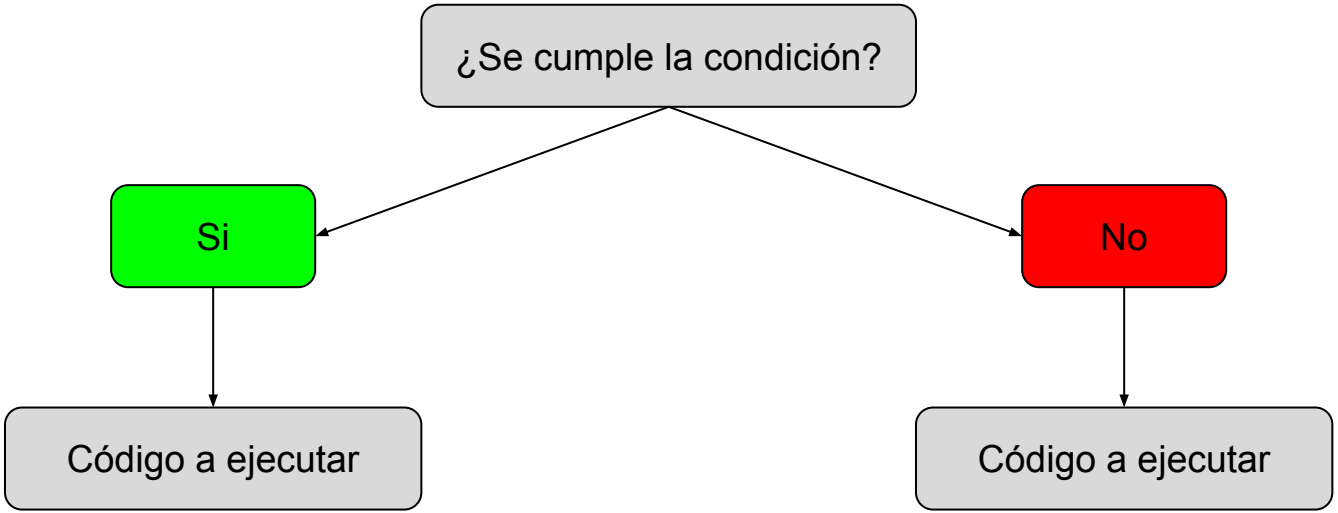
El flujo de ejecución de este programa es secuencial hasta que se ejecuta la instrucción CALL. Tras eso, se ejecutan las instrucciones de la <rutina b> (también de forma secuencial). Una vez terminada esta rutina, se regresa a la rutina que invocó a <rutina b>, y se siguen ejecutando sus instrucciones secuencialmente.

Flujo de la estructura condicional

Q4 añade la posibilidad de utilizar una estructura condicional para ejecutar una parte de código si se cumple una condición dada, caso contrario, se ejecutará otro fragmento de código. Para eso se añaden las siguientes instrucciones:

a. CMP: permite comparar dos valores
b. Salto incondicional o absoluto: se ejecuta **siempre** (JMP)
c. Saltos condicionales o relativos: se ejecutan al cumplirse cierta condición (JGE, JNE, etc...)

Estructura condicional - Funcionamiento



Ejemplo

Se quiere saber si un alumno aprobó una materia de la carrera (se considera aprobado a un alumno con una nota **igual o mayor** a 40 puntos, básicamente, un 4). Para eso se arma el siguiente programa, el cual posee la siguiente documentación:

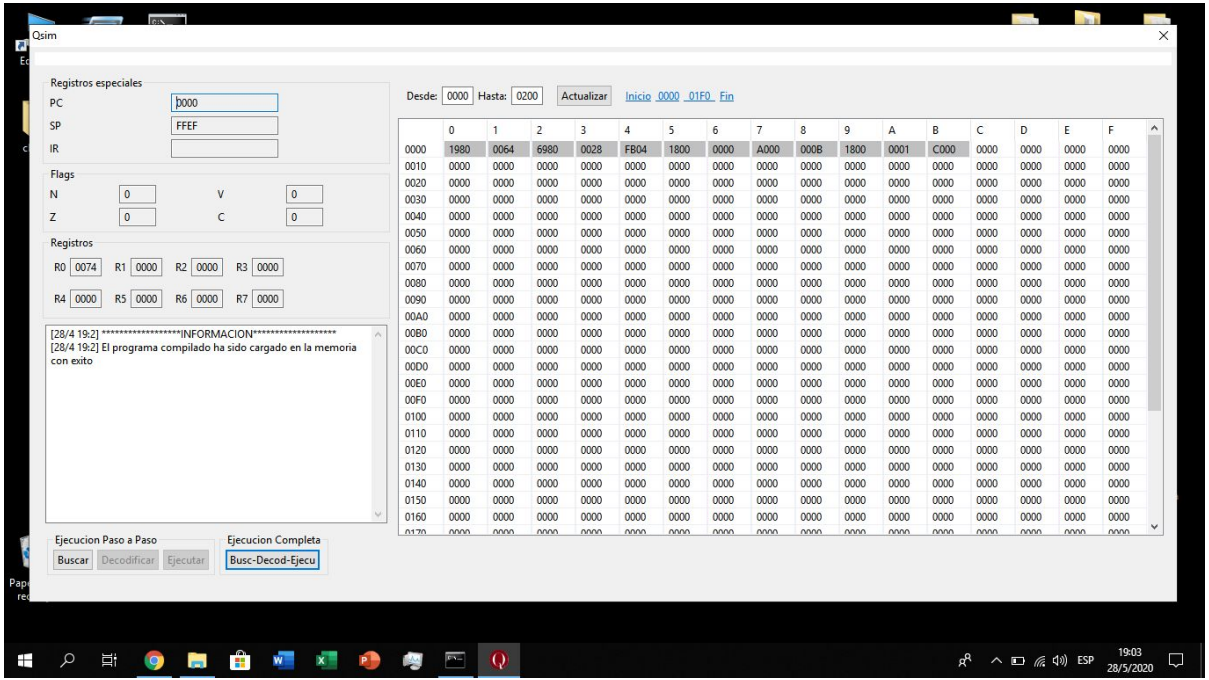
Requiere: en R6 un número en el rango [0..100] en BSS
Modifica: ---
Retorna: En R0 un 1 si aprobó, y 0 en caso contrario

Ejemplo

En este caso, el alumno en cuestión sacó una nota de 100/100, por ende, el valor que le vamos a pasar a R6 es 0064

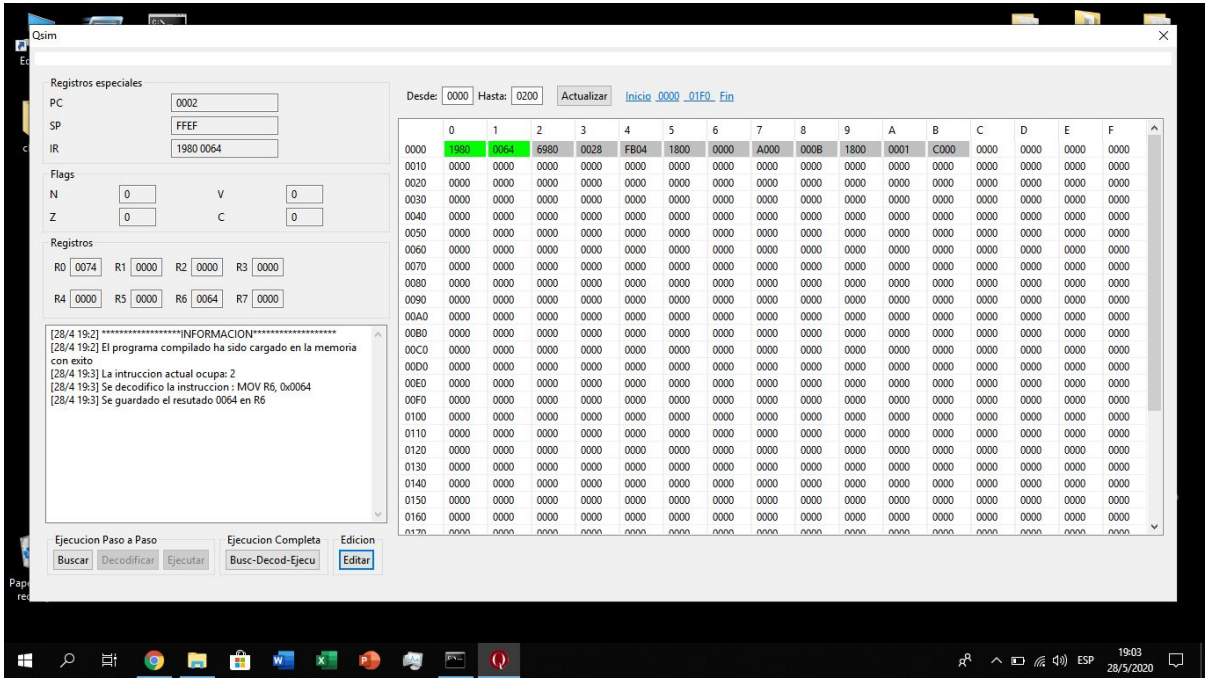
```
MOV R6, 0x0064 // parametrización
    analisisDeNota CMP R6 0x0028
                   JGE materiaAprobada
                   MOV R0, 0x0000
                   JMP fin
materiaAprobada MOV R0, 0x0001
            fin RET
```

En la imagen se puede ver el programa cargado en memoria desde la celda 0x0000 y el valor con el cual empieza PC.



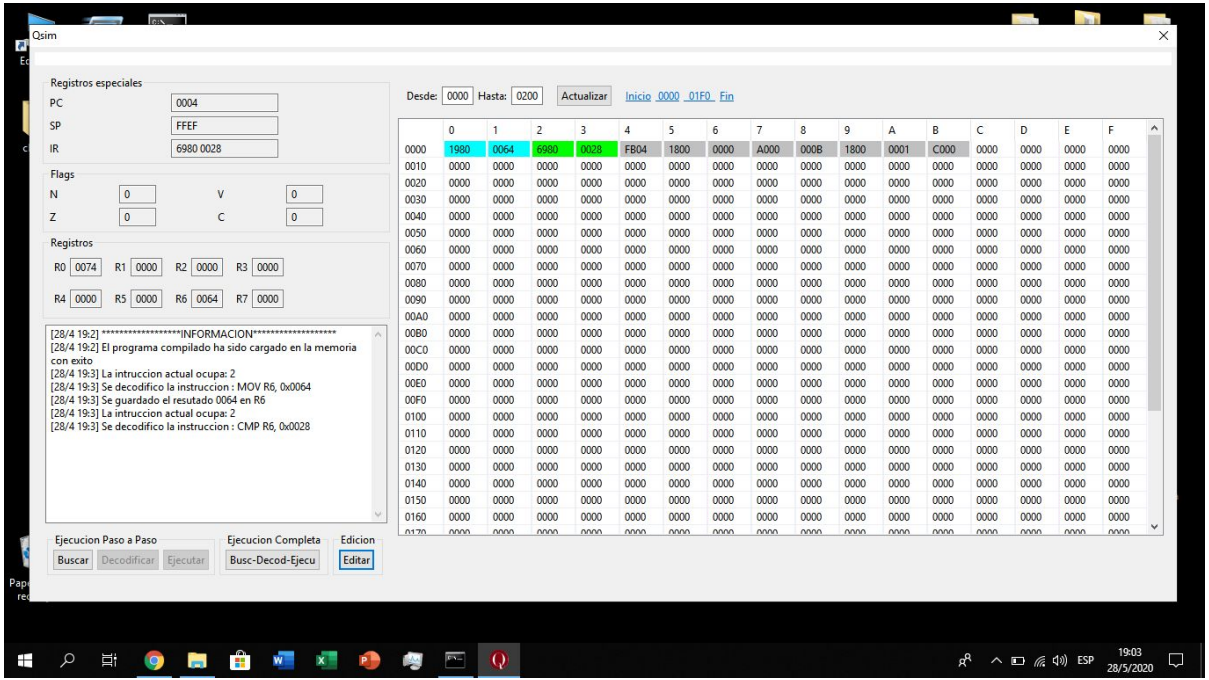
Tras la ejecución de la primer instrucción vemos los siguientes cambios:

a. **PC posee el valor 0002 (cambio secuencial)**
b. IR se actualiza su contenido
c. R6 ahora contiene el valor 0064



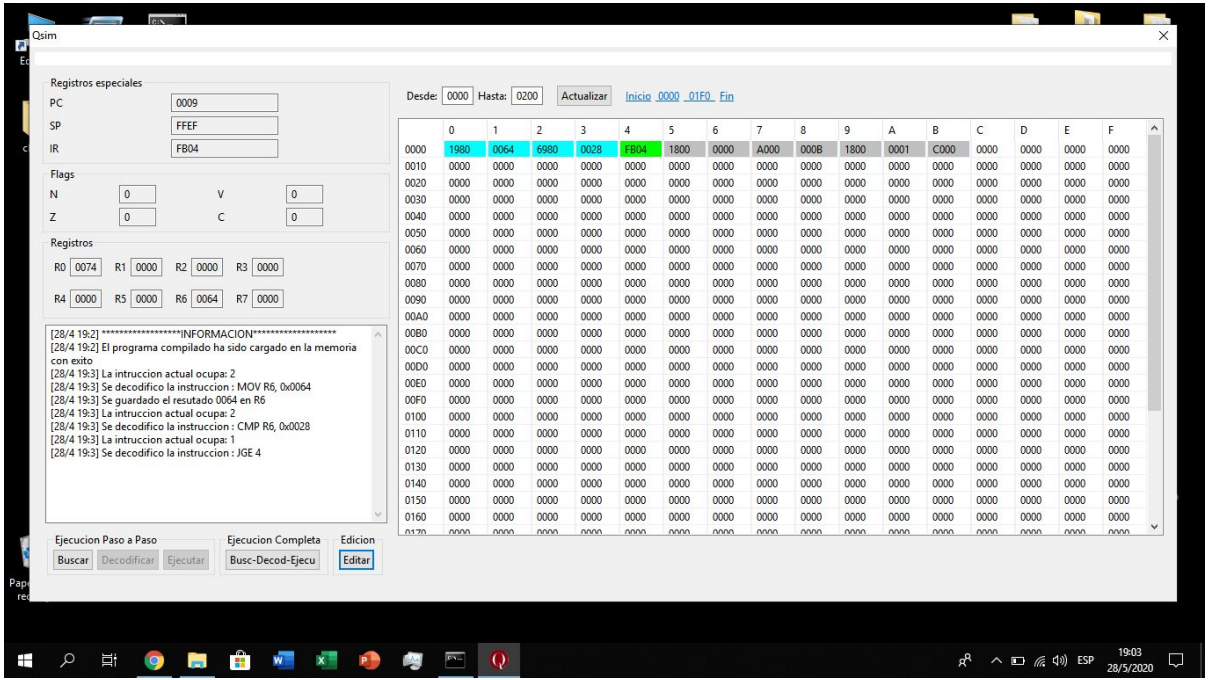
Tras la ejecución de la siguiente instrucción vemos los siguientes cambios:

a. **PC posee el valor 0004 (cambio secuencial)**
b. IR se actualiza su contenido
c. CMP realiza la comparación de los valores



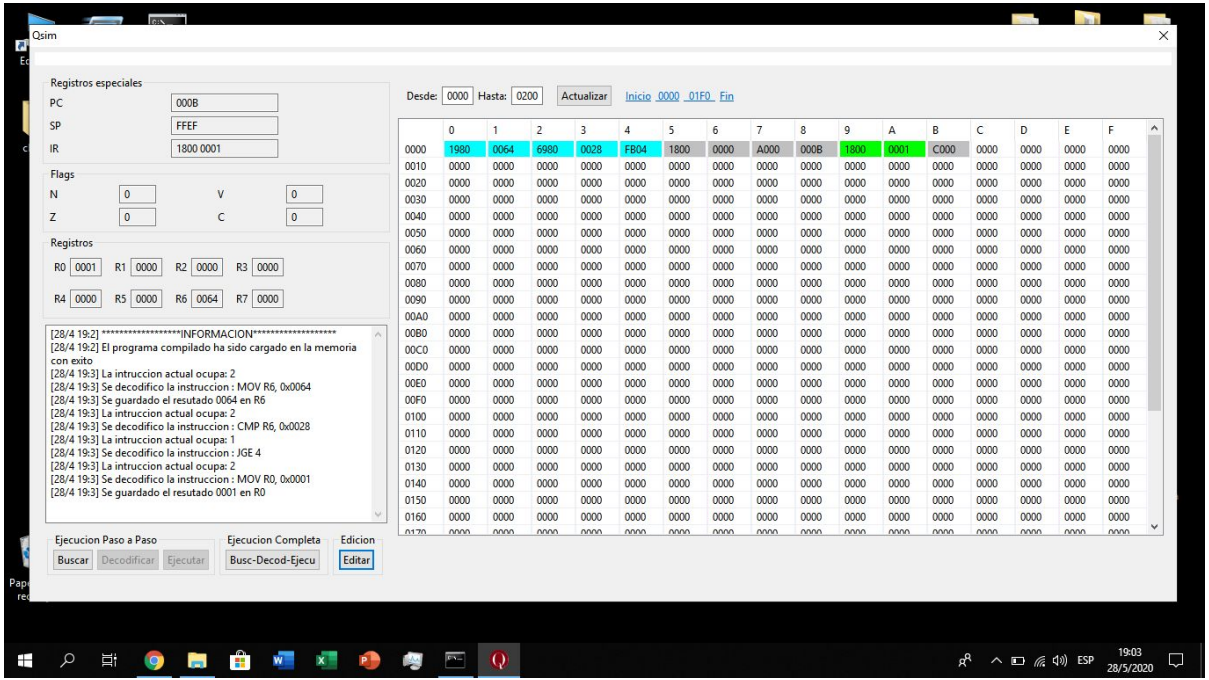
Tras la ejecución de la siguiente instrucción vemos los siguientes cambios:

a. **PC posee el valor 0009 (desvío causado por el salto, a PC 2 se le suma el desplazamiento a realizar)**
b. IR se actualiza su contenido
c. Se verifica si la condición se cumple o no. En este caso se cumple, por ende, se realizará el salto



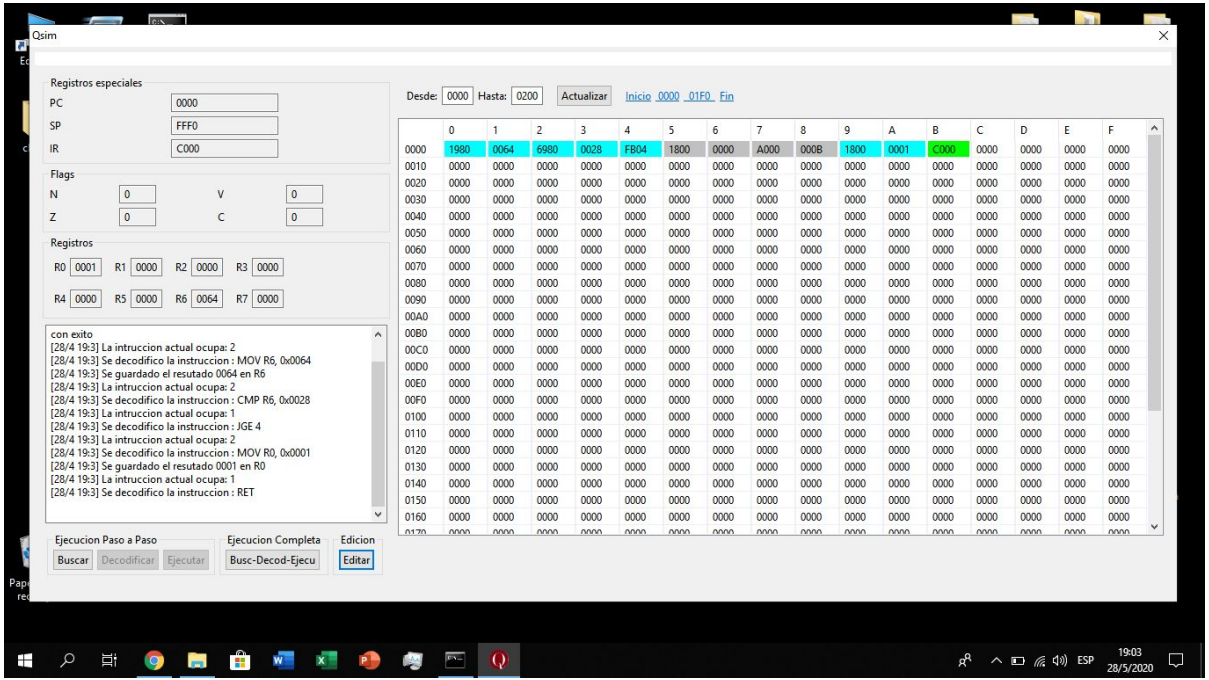
Tras la ejecución de la siguiente instrucción vemos los siguientes cambios:

a. **PC posee el valor 000B (cambio secuencial)**
b. IR se actualiza su contenido
c. R0 actualiza su contenido a 0001



Tras la ejecución de la siguiente instrucción vemos los siguientes cambios:

a. **PC posee el valor 000C (cambio secuencial)**
b. IR se actualiza su contenido
c. Se ejecuta el RET



¿Que hubiese ocurrido en caso de no cumplirse la condición?

En caso de no cumplirse la condición, se habrían seguido ejecutado las instrucciones de manera secuencial, las cuales representaban la porción de código asignada para cuando la condición no se cumpliera.

¿Cómo es el flujo de ejecución en este ejemplo?

El flujo de ejecución de este programa es secuencial hasta el momento en que se ejecuta el salto condicional. Como se cumple la condición, se desvía el flujo hacia la porción de código que debe ejecutarse