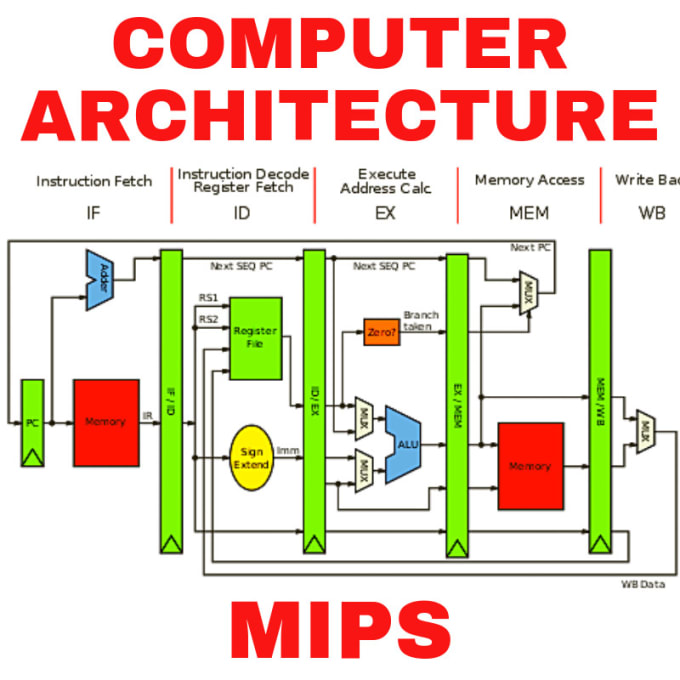
# EC

# ENTREGA 2



Realizado por: Jaime Aznar Espinosa

Índice

[Contenido 2](#_Toc34766240)

[1. Objetivos 3](#_Toc34766241)

[2. Práctica 4 4](#_Toc34766242)

[2.1 Cuestión 5 4](#_Toc34766243)

[2.2 Cuestión 6 5](#_Toc34766244)

[2.3 Cuestión 8 6](#_Toc34766245)

[3. Práctica 5 7](#_Toc34766246)

[3.1 Cuestión 2 7](#_Toc34766247)

[3.2 Cuestión 7 9](#_Toc34766248)

[3.3 Cuestión 10 10](#_Toc34766249)

[4. Práctica 6 11](#_Toc34766250)

[4.1 Cuestión 3 11](#_Toc34766251)

[4.2 Cuestión 7 11](#_Toc34766252)

[4.3 Cuestión 8 13](#_Toc34766253)

[5. Observaciones 14](#_Toc34766254)

[6. Conclusiones 14](#_Toc34766255)

# Objetivos

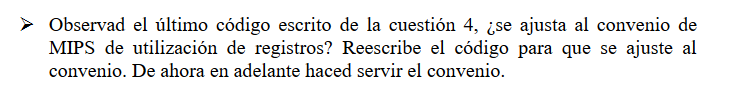
En la práctica 4 el objetivo principal es entender el manejo de los desplazamientos y el efecto aritmético que producen. En concreto multiplicaciones y divisiones enteras mediante sumas y desplazamientos, así como empezar a crear funciones para tener bloques más definidos que realicen funcionalidades concretas.

En la práctica 5 el objetivo es ampliar lo que dejamos en la práctica 4 con los saltos a funciones, añadiendo otras instrucciones con saltos condicionales que rompen la ejecución secuencial del programa, y que pueden derivar en distintos finales, saltando entre etiquetas.

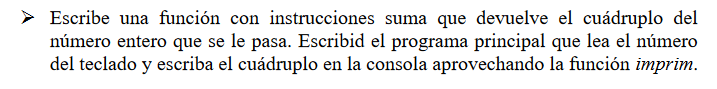
Por último, en la práctica 6 se ahonda más en la distribución de memoria en MIPS con operaciones de lectura y escritura en memoria, así como la definición de datos en memoria y la posibilidad de utilizar funciones de entrada/salida de cadenas de caracteres, no como en las prácticas anteriores que introducíamos un carácter.

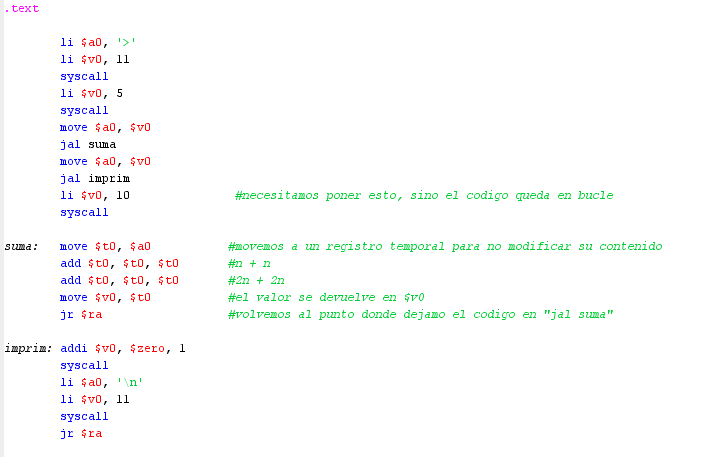
# Práctica 4

## Cuestión 5



La cuestión 4 pedía realizar lo siguiente:





Como se puede apreciar los registros son los adecuados, ya que he utilizado los $a.. para pasar parámetros a las funciones, y el cálculo temporal dentro de funciones ha sido realizado con los registros temporales $t.. .

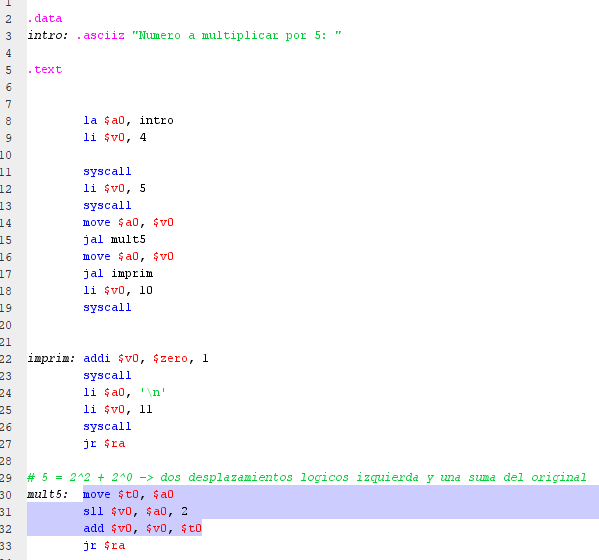
## Cuestión 6



Para realizar multiplicaciones mediante desplazamientos usaremos potencias de 2, mediante el sumatorio de estas mismas de la siguiente manera:

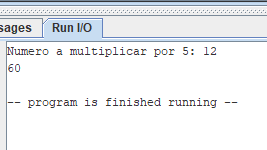
5 = 2^2 + 2^0

O lo que es lo mismo, un desplazamiento lógico a la izquierda de 2 bits y la suma del número original.

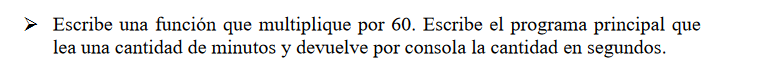


Como se puede ver en la zona remarcada, me guardo el valor original en $t0, realizo el desplazamiento sobre $v0 y después le sumo el valor original en $v0 que será el registro de retorno.

Este código da como lugar el siguiente resultado:



## Cuestión 8



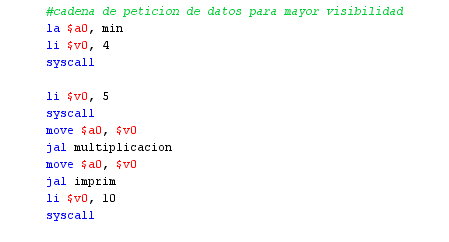
Siguiendo la misma filosofía que para el ejercicio anterior tenemos que hacer la multiplicación por 60, que resulta en:

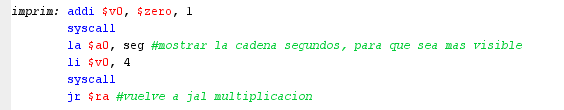
60 = 2^5 + 2^4 + 2^3 + 2^2

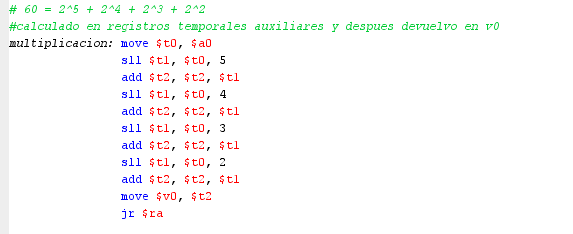
Es decir, la suma de los desplazamientos lógicos izquierda de 5 bits más 4, 3 y 2.

He intentado hacer este código en un bucle, pero he tenido problemas ya que intentaba decrementar un registro y realizar los desplazamientos guiados en este mismo registro, pero la instrucción sll no admite un registro como argumento, sino que tiene que ser un valor inmediato, con lo cual, el código ha quedado así:

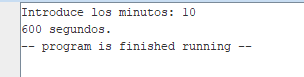






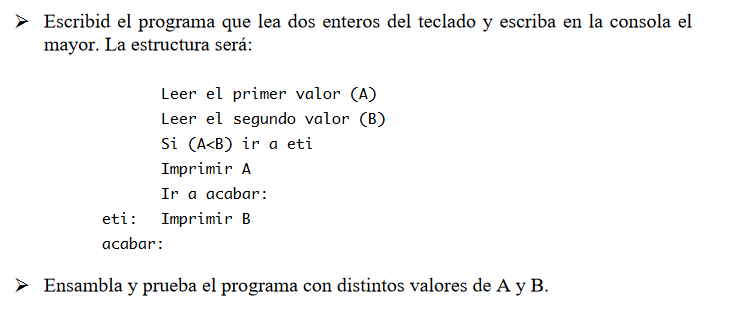


Este código tiene como resultado:

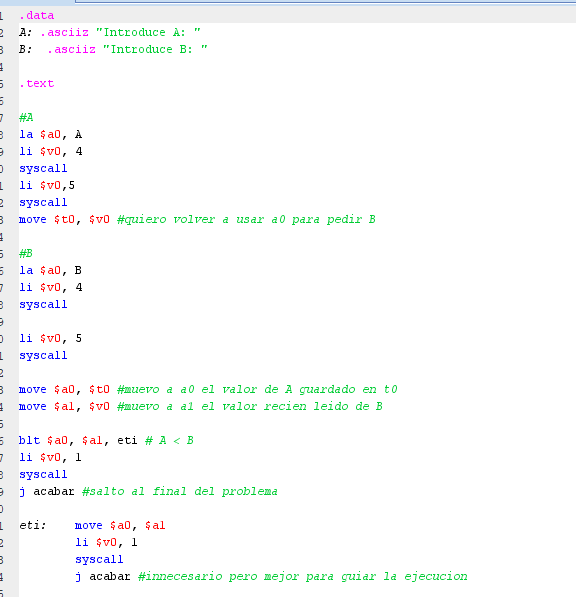
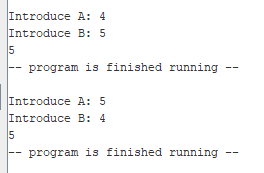


# Práctica 5

## Cuestión 2

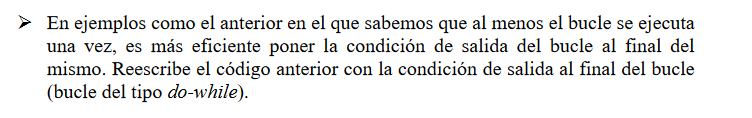


Para este ejercicio tenemos que leer dos valores, realizar una comparación y hacer Branch si b es mayor que a, después desde esta nueva etiqueta saltar a la etiqueta acabar para evitar bucles y también saltar a acabar desde el final del “main” por si no se cumple la condición, ya que sino al estar la etiqueta de impresión de B debajo del “main”, imprimirá ambos valores y después finalizará, el código del ejercicio se encuentra en la siguiente página.



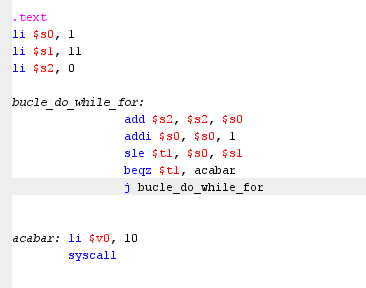
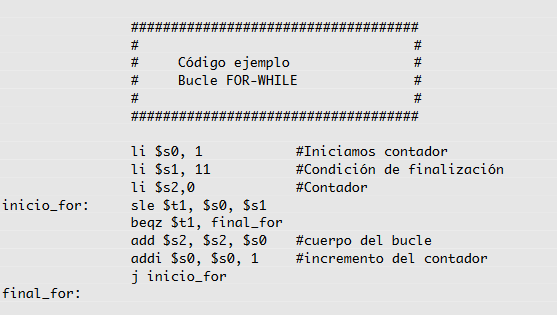
## 

## Cuestión 7

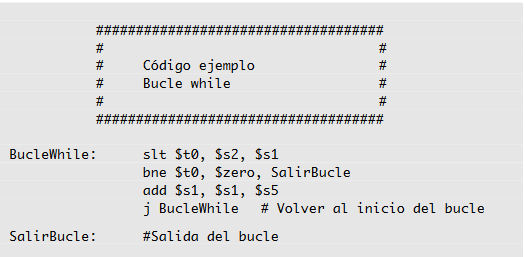
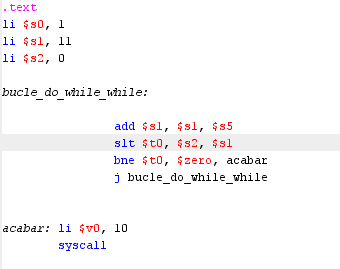


En este caso he decidido reescribir tanto el bucle for a dowhile, como el while previo:

Bucle For:



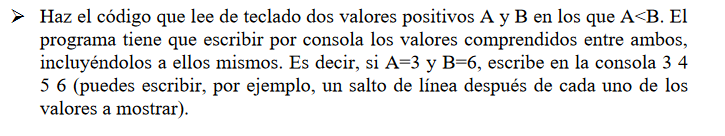
Bucle While:



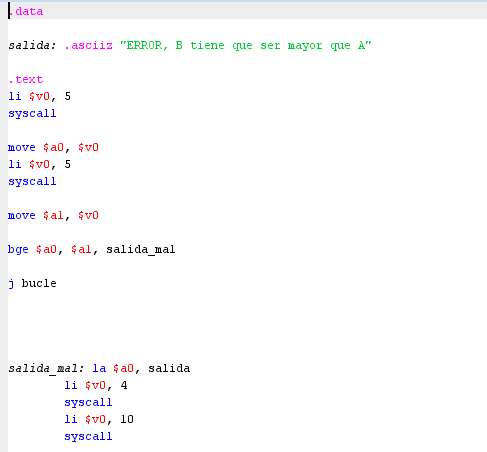
Lo único que hacemos es variar la condición de salida al final del bucle, para que la primera iteración sea obligatoria.

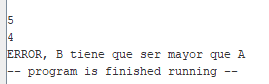
## 

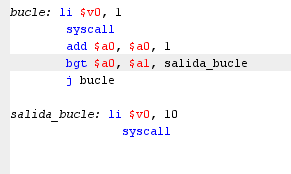
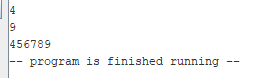
## Cuestión 10



Para este ejercicio he añadido la comprobación de que A sea menor que B, ya que de otra manera el programa no tiene sentido:

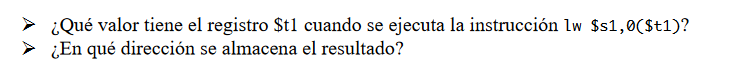


Si A es mayor que B entonces se hace un print con un mensaje de error y acaba el programa, de otro caso, se hace un jump al bucle donde se realizan los prints de los números desde A hasta B inclusive:



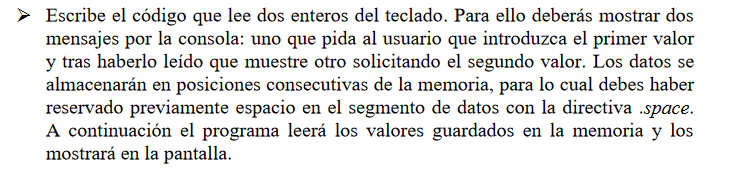
# Práctica 6

## Cuestión 3

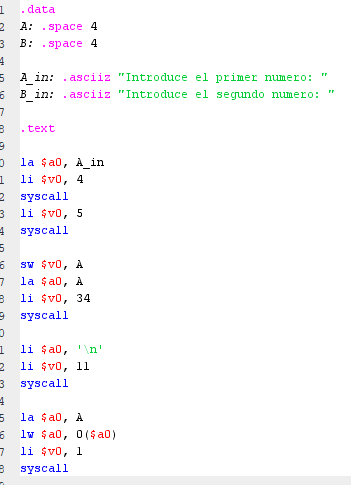


El registro $t1, previamente ha sido cargado con la dirección de memoria de B con la instrucción load address, Con lo cual, al haber definido una variable previa de tipo word, dicha variable previa (A) estará en la primera dirección de memoria del segmento data, es decir 0x10010000 y el valor de B estará en los siguientes 4 bytes(tamaño de palabra), es decir **$t1 será 0x10010004.**

## Cuestión 7

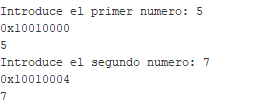


Para guardar dos datos de tipo palabra en un lugar consecutivo en memoria, tendremos que usar la directiva .space con 4 bytes que es el tamaño de la palabra. Después de leer por pantalla, usaremos sw para guardar el valor de $v0 en las etiquetas vacías que hemos declarado en data y haremos un print del valor leído y de la dirección de memoria en la que está. El código queda como sigue:



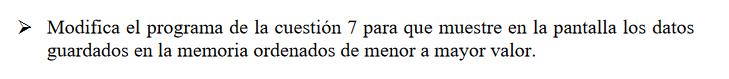
El código es exactamente igual para el segundo número.

La salida es la siguiente:



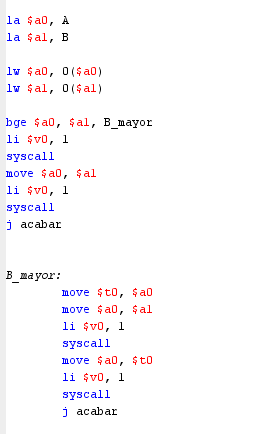
Como podemos ver, el .data empieza en 0x10010000 que es donde se aloja A e inmediatamente después almacenamos B.

## Cuestión 8

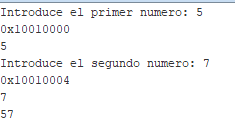


Lo que tenemos que hacer para realizar la comparación de los dos valores y colocarlos en orden, es cargar ambas etiquetas en registros, acceder a la palabra que contienen, y compararlos, si la comparación es satisfactoria se redirigirá a una etiqueta donde los imprimirá de una manera, y si no la cumple, continuará el hilo de ejecución hasta que se acabe.

El código extra del programa anterior es el siguiente:



La etiqueta acabar simplemente finaliza la ejecución con li $v0,10 syscall.



# 

# Observaciones

En varios ejercicios, he realizado la llamada a etiquetas condicionales pasando los parámetros en los registros $a en lugar de los registros $s por comodidad a la hora de imprimir, ya que la mayor parte de estas etiquetas se basaban en imprimir en un orden en lugar del predefinido, con lo que resultaba más fácil de esta manera.

# Conclusiones

La práctica ha servido para afianzar los conocimientos previos, además hemos aprendido el uso de etiquetas para realizar funciones específicas así como el uso del convenio de registros para tener un uso adecuado y ágil. En cuanto a las operaciones aritméticas hemos profundizado más con las divisiones y las multiplicaciones mediante desplazamientos. Como hemos comentado antes, el uso de etiquetas ha sido algo primordial en esta entrega, ya que mediante éstas hemos realizado bucles y código condicional, cosas muy importantes en todo lenguaje de programación. Por último, hemos empezado a conocer la memoria de mips, tanto el acceso como la escritura en variables definidas en el segmento data, la escritura y lectura en lugares consecutivos en memoria, que nos será útil a la hora de crear vectores y recorrerlos de palabra media palabra o byte en prácticas posteriores.

