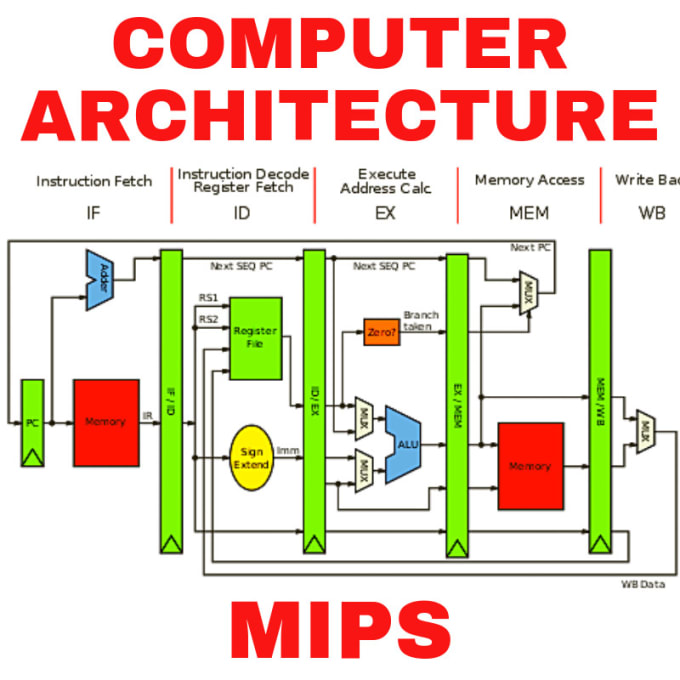
EC

ENTREGA 2



Realizado por: Jaime Aznar Espinosa

Índice

[1. Objetivos 3](#_Toc34557203)

[2. Práctica 4 3](#_Toc34557204)

[2.1 Cuestión 5 3](#_Toc34557205)

[2.2 Cuestión 6 4](#_Toc34557206)

[2.3 Cuestión 8 4](#_Toc34557207)

[3. Práctica 5 5](#_Toc34557208)

[3.1 Cuestión 2 6](#_Toc34557209)

[3.2 Cuestión 7 7](#_Toc34557210)

[3.3 Cuestión 10 8](#_Toc34557211)

[4. Práctica 6 9](#_Toc34557212)

[4.1 Cuestión 3 9](#_Toc34557213)

[4.2 Cuestión 7 9](#_Toc34557214)

[4.3 Cuestión 8 11](#_Toc34557215)

[5. Conclusiones 12](#_Toc34557216)

# Objetivos

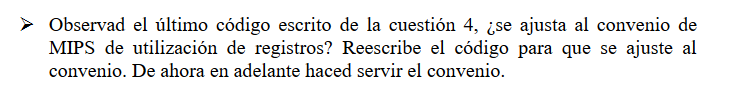
En la práctica 4 el objetivo principal es entender el manejo de los desplazamientos y el efecto aritmético que producen. En concreto multiplicaciones y divisiones enteras mediante sumas y desplazamientos, así como empezar a crear funciones para tener bloques más definidos que realicen funcionalidades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

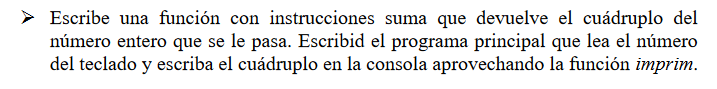
Por último, en la práctica 6 se ahonda más en la distribución de memoria en MIPS con operaciones de lectura y escritura en memoria, así como la definición de datos en memoria y la posibilidad de utilizar funciones de entrada/salida de cadenas de caracteres, no como en las prácticas anteriores que introducíamos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

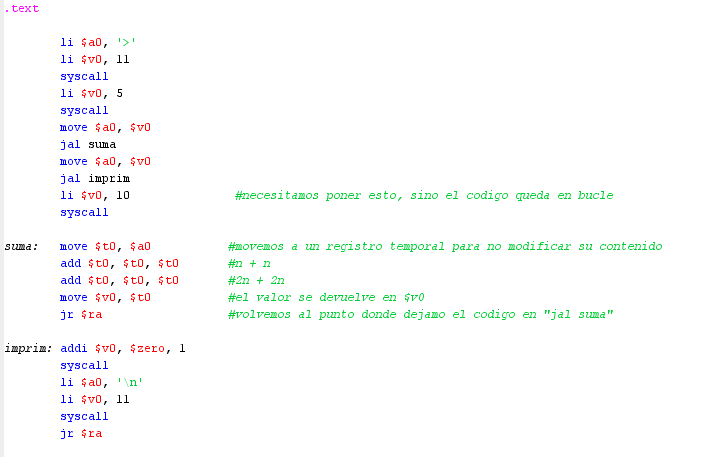
# Práctica 4

## 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.. .

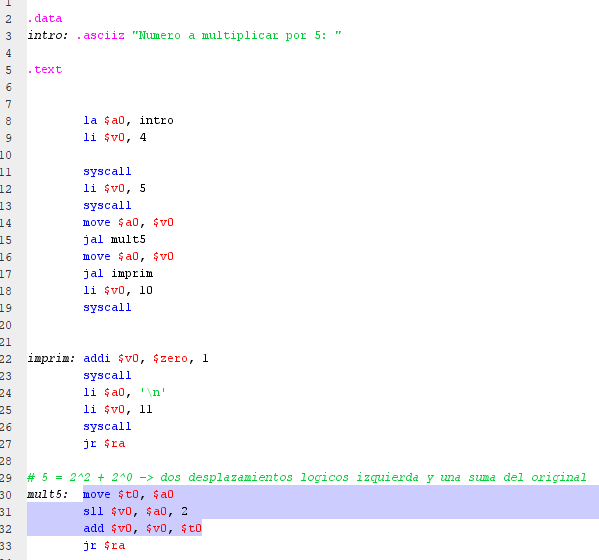
## Cuestión 6



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

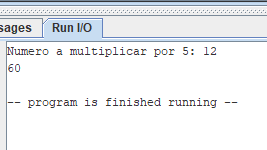
5 = 2^2 + 2^0

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

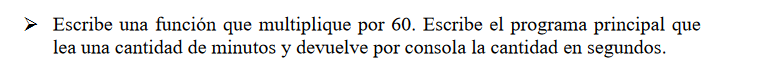


Como se puede ver en la zona remarcada, me guardo el valor original en $t0, realizo el desplazamiento sobre $v0 y después le sumo el valor original en $v0 que será el registro de retorno.

Este código da como lugar el siguiente resultado:



## Cuestión 8



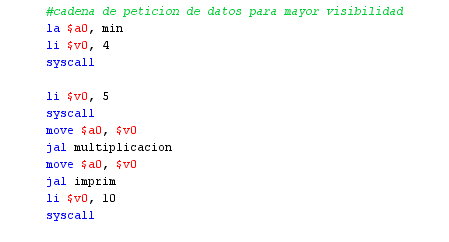
Siguiendo la misma filosofía que para el ejercicio anterior tenemos que hacer la multiplicación por 60, que resulta en:

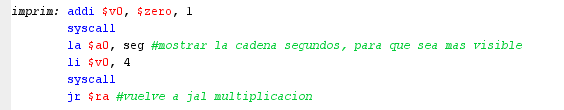
60 = 2^5 + 2^4 + 2^3 + 2^2

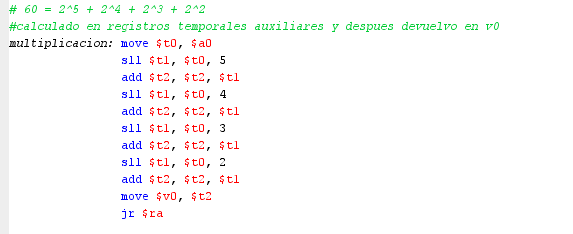
Es decir, la suma de los desplazamientos lógicos izquierda de 5 bits más 4, 3 y 2.

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:

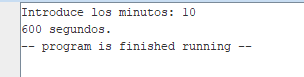






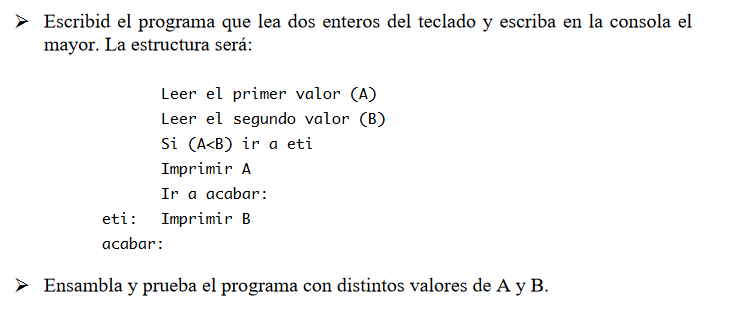


Este código tiene como resultado:

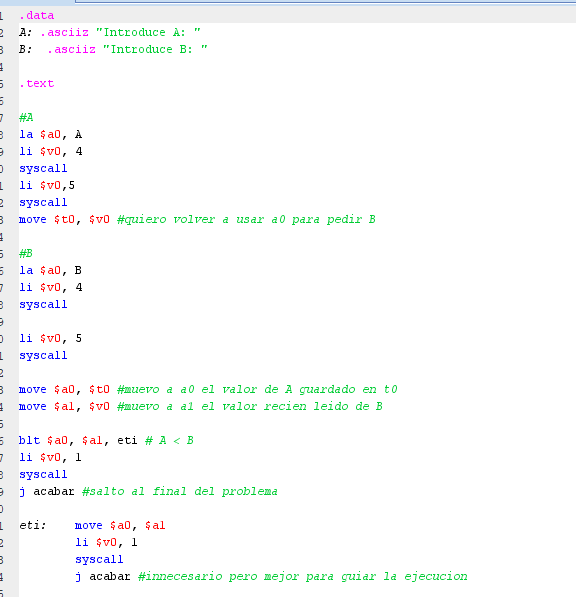
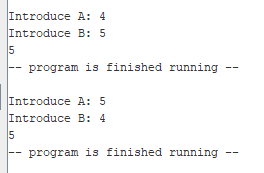


# Práctica 5

## Cuestión 2

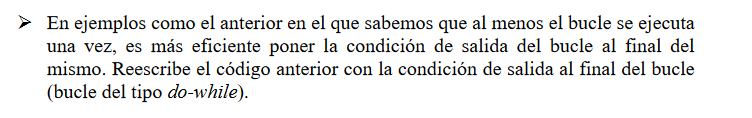


Para este ejercicio tenemos que leer dos valores, realizar una comparación y hacer Branch si b es mayor que a, después desde esta nueva etiqueta saltar a la etiqueta acabar para evitar bucles y también saltar a acabar desde el final del “main” por si no se cumple la condición, ya que sino al estar la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



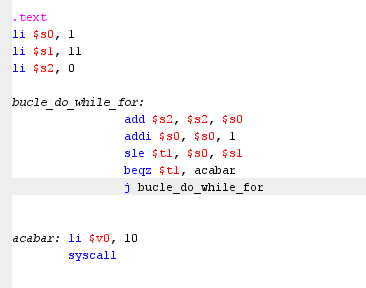
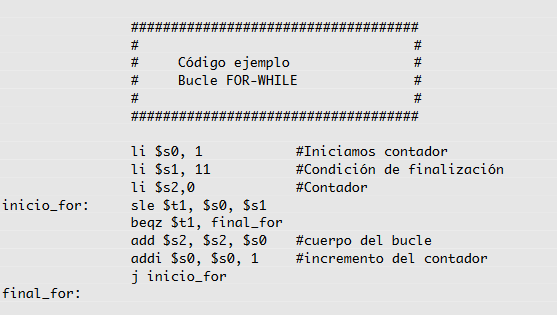
## 

## Cuestión 7

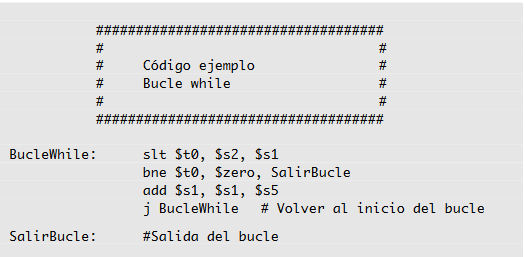
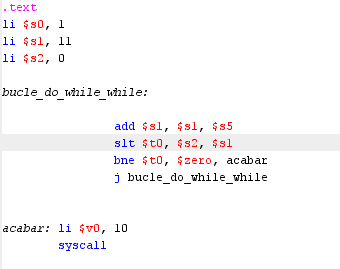


En este caso he decidido reescribir tanto el bucle for a dowhile, como el while previo:

Bucle For:

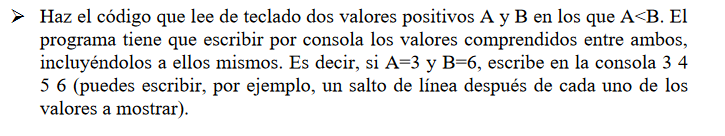


Bucle While:

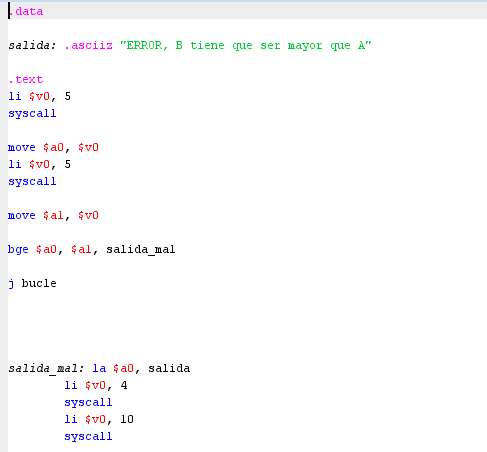


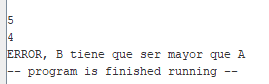
Lo único que hacemos es variar la condición de salida al final del bucle, para que la primera iteración sea obligatoria.

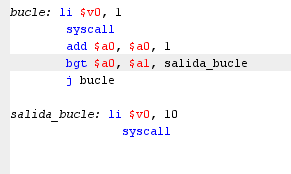
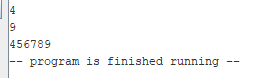
## Cuestión 10



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

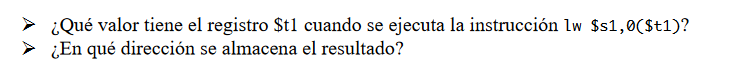


Si A es mayor que B entonces se hace un print con un mensaje de error y acaba el programa, de otro caso, se hace un jump al bucle donde se realizan los prints de los números desde A hasta B inclusive:

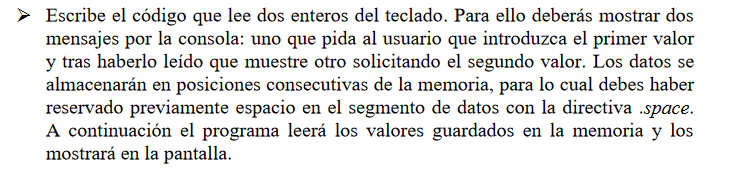


# Práctica 6

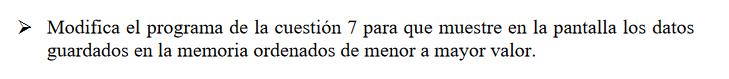
## Cuestión 3



## Cuestión 7



## Cuestión 8



# Conclusiones

