1.- Escribe un programa en ensamblador ARM que dado un vector **datos** de números enteros, de longitud **tam**, escriba en **resul** un vector de igual tamaño con los valores del vector **datos** pero en orden inverso (es decir, el último elemento de **datos** deberá aparecer el primero en **resul** y el primero de **datos** el último en **resul**). A continuación, te presentamos un segmento de datos que te podría servir como base.

```
.data
tam: .word 8
datos: .word 8,3,4,2,5,7,3,1
resul: .word 0,0,0,0,0,0,0,0
.text
.global main
main:
```

2.- Escribe una función en ensamblador ARM que cuente el número de veces que

2.- Escribe una función en ensamblador ARM que cuente el número de veces que aparece un determinado valor "**n**" en un vector datos. La función se debe llamar **hist** y debe aceptar tres parámetros de entrada:
- el primero, por el registro **r0**, indicará el valor **n** a buscar en el vector
- el segundo, por el registro **r1**, indicará la dirección de memoria donde comienza el vector que contiene los datos
- y el tercero, por el registro **r2**, indicará la longitud (número de elementos) del vector.

La función devolverá por el registro r0 el número de veces que aparece "n" (indicado en r0) en el vector.
A continuación, aparece un código de ejemplo que llama a la función que debes implementar (solo debes implementar la función).

```
.data
tamdat: .word 15
datos: .word 2, 4, 2, 8, 2, 4, 2, 7, 5, 6, 5, 3, 7, 6, 8
res: .word 0

.text
.global main
main:
   push {lr}          @ salvamos reg. lr pues se va a perder cuando
                      @ se llame a alguna función (bl)
   ldr r1, =datos     @ cargamos en r1 la dirección del vector
   ldr r2, =tamdat
   ldr r2, [r2]       @ cargamos en r2 el tamaño del vector
   mov r0, #4         @ cargamos en r0 el número a buscar (4)
   bl hist            @ llamamos a la función hist
   ldr r1, =res
   str r0, [r1]       @ almacenamos en memoria (res) el resultado (r0)
   pop {lr}           @ recuperamos reg. lr para poder salir del prog
   mov pc, lr         @ retornamos del programa principal

hist: ...
```

3.- Escribe un programa en ensamblador ARM que usando la función creada en el problema anterior (**hist**), calcule el número de veces que aparecen cada uno de los valores almacenados en el vector **valores** dentro del vector **datos** y lo almacene en el vector **res**. Es decir, en **res**[i] deberás almacenar el número de veces que aparece **valores**[i] en el vector **datos** (para i=0 hasta **tamval** -1). El tamaño del vector valores se especifica en la variable **tamval** y el tamaño del vector datos se especifica en la variable **tamdat**.

Para el ejemplo siguiente, el vector **res** al finalizar debería contener los valores: 4, 2, 2, 0.

```
.data
tamval: .word 4
tamdat: .word 15
datos: .word 2, 4, 2, 8, 2, 4, 2, 7, 5, 6, 5, 3, 7, 6, 8
valores: .word 2, 4, 6, 9
res: .word 0, 0, 0, 0

.text
main: ...

hist: ...
```

Primer examen parcial, abril 2019

Nombre:	DNI:

1. Escribe un programa en ensamblador ARM que dado un vector de números enteros (con número de elementos **numel**), escriba en **resultado** la diferencia entre el valor máximo y mínimo del vector. En el ejemplo, resultado = máximo(vector)-mínimo(vector) = 9 – (-7) = 16.
   El fichero con el programa **debe llamarse ej1.s**

```
.data
numel:      .word 8
vector:     .word 8,-3,4,-7,9,-7,6,-1
resultado:  .word 0

.text
.global main
main:
            ...
```

El código debe funcionar con **el ejemplo anterior** y también con los **siguientes 3 ejemplos**

```
numel:      .word 1
vector:     .word 3
```

En este caso, resultado = máximo(datos)-mínimo(datos) = 3 – (-3) = 0.

```
numel:      .word 0
vector:     .word 0
```

En este caso, resultado = 0

```
numel:      .word 5
vector:     .word -1,-2,-3,-4,-5
```

En este caso, resultado = máximo(datos)-mínimo(datos) = (-1) – (-5) = 4.

2. Escribe una **función** en ensamblador ARM que dado un vector de números enteros devuelva la suma de los valores absolutos de los elementos del vector:

$$\sum_{i=0}^{numel-1} |vect[i]|$$

La función se debe llamar **abssum,** y aceptará los siguientes parámetros de entrada:

- por el registro **r0** la dirección del vector
- por el registro **r1** el número de elementos de dicho vector

La función devolverá el resultado (suma de los valores absolutos) por el registro **r0**. La función **DEBERÁ** cumplir el convenio de llamada a funciones.

Para calcular el valor absoluto de un número **debéis** usar la función **abs** que **se os da ya implementada**, la cual acepta el número natural por el registro **r0** y devuelve su valor absoluto por el registro **r0** ($r0 = |r0|$).

Ni el main ni la función **abs** pueden ser cambiados. Para probar la función podéis usar el código que aparece a continuación (el resultado del ejemplo es 36). No olvidéis que se trata de un ejemplo, y la implementación tanto de la función **abs** como del **main** podría ser cualquier otra.
El fichero con el programa **debe llamarse ej2.s**

```
.data
mynumel:    .word 8
myvect:     .word 8,-3,4,-2,5,7,6,1
myres:      .word 0

.text
.global main
main:       push {lr}
            ldr r0, =myvect
            ldr r1, =mynumel
            ldr r1, [r1]
            bl abssum
            ldr r1, =myres
            str r0, [r1]
            pop {lr}
            mov pc, lr

abs:        mov r1, #0
            cmp r0, r1
            sublt r0, r1, r0
            mov pc, lr

abssum:
            ...
```

El código debe funcionar con **el ejemplo anterior** y también con los **siguientes 2 ejemplos**:

```
mynumel:    .word 1
myvect:     .word -100
```

En este caso, r0 = 100

```
mynumel:    .word 0
myvect:     .word 0
```

En este caso, r0 = 0

Problema 1 programa

Escribe un programa en ensamblador ARM que dado un vector **datos** de números enteros y de longitud **len** escriba en **res** la amplitud del rango de los números almacenados en dicho vector. Es decir, debe escribir en **res** la diferencia entre el valor máximo y mínimo del vector (en el ejemplo, res = máximo(datos)-mínimo(datos) = 9 – (-7) = 16).

Para probar el código podéis usar el vector que aparece a continuación. No olvidéis que se trata de un ejemplo, y que el programa debería funcionar para cualquier tipo de vector (de cualquier tamaño y con distintos números enteros).

El fichero con el programa **debe llamarse ej1.s**

```
.data
len:        .word 8
datos:      .word 8,-3,4,-7,9,-7,6,-1
res:        .word 0

.text
.global main
main:
```

Problema 2 funcion

Escribe una **función** en ensamblador ARM que dado un vector de números enteros devuelva el máximo de los valores absolutos de los

Problema 2 funcion

Escribe una **función** en ensamblador ARM que dado un vector de números enteros devuelva el máximo de los valores absolutos de los elementos del vector:

MAX(|vect(i)|)

La función se debe llamar **absmax,** y aceptará los siguientes parámetros de entrada:

- por el registro **r0,** la dirección del vector
- por el registro **r1,** la longitud de dicho vector

La función devolverá el resultado (máximo de los valores absolutos) por el registro **r0**, y **DEBERÁ** cumplir el convenio de llamada a funciones.

Para calcular el valor absoluto de un número **debéis** usar la función **abs** que **se os da ya implementada**, la cual acepta el número natural por el registro **r0** y devuelve su valor absoluto por el registro **r0** (r0 = |r0|).

Para probar la función podéis usar el código que aparece a continuación. No olvidéis que se trata de un ejemplo, y la implementación tanto de la función **abs** como del **main** podría ser cualquier otra.

El fichero con el programa **debe llamarse ej2.s**

```
.data
mytam:      .word 8
myvect:     .word 8,-3,4,-2,5,7,6,1
myres:      .word 0

.text
.global main
main:       push {lr}
            ldr r0, =myvect
            ldr r1, =mytam
            ldr r1, [r1]
            bl absmax
            ldr r1, =myres
            str r0, [r1]
            pop {lr}
            mov pc, lr

abs:        mov r1, #0
            cmp r0, r1
            sublt r0, r1, r0
            mov pc, lr

absmax:
```

Problema 4 funcion

Problema 4 funcion

Escribe una **función** en ensamblador ARM que dado un vector de números naturales devuelva el número almacenado en dicho vector que tenga más "1" en su representación binaria. Tu función se debe llamar **maxones** y aceptará los siguientes parámetros de entrada: por el registro **r0** la dirección del vector y por el registro **r1** la longitud de dicho vector. La función devolverá el resultado por el registro **r0** y **DEBERÁ** cumplir el convenio de llamada a funciones.

Para saber el número de unos de la representación binaria de un número debéis usar la función **ones** que **se os da ya implementada**, la cual acepta el número a chequear por el registro **r0** y devuelve también por el registro **r0** el número de "1" de la representación binaria del valor de entrada.

Para probar la función podéis usar el código que aparece a continuación. No olvidéis que se trata de un ejemplo, y la implementación tanto de la función **ones** como del **main** podría ser cualquier otra.

El fichero con el programa **debe llamarse ej4.s**

```
.data
tam:            .word 8
vect:           .word 64,25,9,23,56,77,87,100
res:            .word 0
.text
.global main
main:    push {lr}
         ldr r0, =vect          @ Cargamos direccion del vector
en r0
         ldr r1, =tam
```

```
         ldr r1, [r1]           @ Cargamos longitud del vector
 en r1
         ldr r4, =res           @ Cargamos la direccion de res
 en r4
         bl maxones             @ Invocamos funcion
         str r0, [r4]           @ Guardamos el resultado en res
         pop {pc}               @ Finalizamos el programa

 ones:   @Codigo de la funcion que devuelve el numero de 1s en
 la
         @representación binaria de un valor de entrada.
         @(Hay que llamarla desde maxones y no se puede
modificar)
               mov r2, #0
         onesloop:
               and r3, r0, #1
               add r2, r2, r3
               lsrs r0, #1
               bne onesloop
               mov r0, r2
               bx lr

maxones:
```