**Ejercicios**

**Ejercicio 1**

Realizar un programa que sume dos números de 64 bits cada uno, guardando el resultado en R5 R6 (64bits), los números a sumar estarán dispuesto en R1 R2(64bits) y R3 R4(64bits) respectivamente.

ldr r3,SUMA11

ldr r4,SUMA21

adds r1,r3,r4

ldr r3,SUMA12

ldr r4,SUMA22

adc r2,r3,r4

loop: b loop

/\* ==============================

\* CONSTANTES

\* ==============================

\*/

SUMA11: .word 0x1231AAAA

SUMA12: .word 0xB3423433

SUMA21: .word 0xFF232323

SUMA22: .word 0x23323111

**Ejercicio 2**

Realizar un programa que analizando un bit en el registro r3 active una alarma si este bit es 0, el nro de bit a analizar está guardado en el registro r4, la alarma se activa escribiendo un 1 en el bit 16 de r6.

V1:

mov r3,#0x7F

mov r3,#0100

mov r3,#0B100

mov r4,#7 @ verificar el bit 7

mov r0,#0x01

lsl r0,r4

and r3,r0

cmp r3,#0

bne seguir

orr r6,#(1<<16)

seguir:

v2:

reset:

mov r3,#0x7F

mov r4,#7 @ verificar el bit 7

mov r0,#0x01

tst r3,r0,lsl r4

orreq r6,r6,#(1<<16)

**Ejercicio 3**

Realizar una serie de subrutinas que realicen diferentes comparaciones, devolviendo r0 = 0 si estas comparaciones fueron falsas y r0<>0 si fueron verdaderas, las comparaciones son las siguientes.

* r1<100 y r1>20
* r1<100 o r2>20
* r1 = 10 o r1 = 15 o r1 = 20
* r1 = 10 y r2 = 15 y r3 = 20

mov r1,#10

mov r2,#15

mov r3,#20

bl comp1

bl comp2

bl comp3

bl comp4

loop: b loop

@ -----------------------------------------------------

@ r1<100 y r1>20

comp1: mov r0,#0 @NO

cmp r1,#100

bhs comp1\_no

cmp r1,#20

ble comp1\_no

mov r0,#1 @SI

comp1\_no:

mov pc,lr

@ -----------------------------------------------------

@ r1<100 o r2>20

comp2: mov r0,#1 @ SI

cmp r1,#100

blo comp2\_si

cmp r2,#20

bhi comp2\_si

mov r0,#0 @ NO

comp2\_si:

mov pc,lr

@ -----------------------------------------------------

@ r1=10 o r1=15 o r1=20

comp3: mov r0,#1 @ SI

cmp r1,#10

beq comp3\_si

cmp r1,#15

beq comp3\_si

cmp r1,#20

beq comp3\_si

mov r0,#0 @ NO

comp3\_si:

mov pc,lr

@ -----------------------------------------------------

@ r1=10 y r2=15 y r3=20

comp4: mov r0,#0 @ NO

cmp r1,#10

bne comp4\_no

cmp r2,#15

bne comp4\_no

cmp r3,#20

bne comp4\_no

mov r0,#1 @ SI

comp4\_no:

mov pc,lr

.end

**Ejercicio 4**

Realizar un programa que dado un numero guardado en R1 lo multiplique por 10, de dos formas distintas, usando la instrucción mul y por corrimiento.

/\* Multiplicar dos numeros usando la instruccion mul \*/

mov r2,#123

mov r1,#10

mul r3,r2,r1

/\* Multiplicar un numero por 10 usando corrimiento \*/

mov r2,#123

mov r3,r2, lsl #1

add r3,r3,r3, lsl #2

**Ejercicio 5**

Realizar un programa que dado un numero guardado en R1 y otro en R2, calcule el resultado de elevar R1 a la potencia de R2 y guardar el resultado en R3 (R3 = R1^R2).

mov r2,#123

mov r1,#3

mov r3,#1 /\* el resultado lo inicializo con 1 \*/

cmp r1,#0 /\* si la potencia es 0 salgo \*/

beq salir

otro1: mul r3,r2,r3 /\* multiplica la base por el resultado \*/

subs r1,r1,#1

bne otro1

**Ejercicio 6**

Realizar un programa que sume los primeros 100 números naturales.

/\* Metodo 1 \*/

mov r1,#99

mov r2,#0

otro1: add r2,r1

subs r1,#1

bne otro1

/\* Método 2 \*/

mov r1,#0

mov r2,#0

otro2: add r2,r1

add r1,#1

cmp r1,#100

bne otro2

**Ejercicio 7**

Realizar un programa que dada una cadena con terminación nula guardada en VECT, la pase a mayúscula, guardando el resultado en el mismo vector VECT.

nota:la cadena de entrada solo contendrá valores alfabéticos o espacio en blanco ('a' - 'z','A' - 'Z').

mov r1,#0

mov r2,#VECT

otro: ldrb r3,[r2]

cmp r3,#0

beq salir

cmp r3,#'a'

subhs r3,r3,#('a'-'A')

strb r3,[r2]

add r2,r2,#1

b otro

salir:

loop: b loop

/\* ==============================

\* CONTANTES

\* ==============================

\*/

VECT: .asciz "hoLa mundo"

.balign 4

.end

**Ejercicio 8**

Realizar un programa que dado un vector de 16 halfword con signos, guarde en R1 el resultado de su promedio.

mov r1,#0

mov r2,#VECT

mov r3,#16

otro: ldrsh r4,[r2],#2

add r1,r1,r4

subs r3,r3,#1

bne otro

mov r1,r1, asr #4

strh r1,PROM

loop: b loop

/\* ==============================

\* CONTANTES

\* ==============================

\*/

VECT: .hword 10,20,30,40,50,60,70,80,90,100,110,120,130,140,-10,-20

PROM: .hword 0

.balign 4

.end

**Ejercicio 9**

Realizar un programa que sume los word de un vector VECTW sin signo en las posiciones indicadas por un segundo vector VECTB de 20 byte,el resultado guardarlo en R1.

ejemplo

VECTB = 1,3,2,0,2......

VECTW = 1,111,222,333,444,555,666,......

se suman 111+333+222+1+222...

mov r1,#0

mov r2,#VECTB

mov r3,#VECTW

mov r4,#20

otro: ldrb r5,[r2],#1

ldr r6,[r3,r5,lsl #2]

add r1,r1,r6

subs r4,r4,#1

bne otro

loop: b loop

/\* ==============================

\* CONTANTES

\* ==============================

\*/

VECTB: .byte 0,5,4,9,2,5,4,6,3,9,8,7,9,4,5,2,1,2,3,8

.balign 4

VECTW: .word 10,20,30,40,50,60,70,80,90,100

.end

**Ejercicio 10**

Dado un vector de 16 words con signo, realizar un programa que encuentre el elemento del vector mas cercano a la media del mismo.

reset: ldr sp,=STACK

ldr r1,=VECTOR

mov r2,#16

mov r3,#0

mov r4,#0

mov r5,#0

mov r6,#0

otro: ldr r0,[r1],#4 @ suma todo los elementos en r3

add r3,r0

subs r2,#1

bne otro

mov r3,r3, asr #4 @ calcula promedio

ldr r1,=VECTOR

mov r2,#16

mov r7,r1

otro2: ldr r0,[r1]

sub r5,r0,r3 @ calcula r5 = abs(r0-r3)

cmp r5,#0

rsbmi r5,#0

cmp r7,r1 @ carga como menor al primer valor del vector

beq primero

cmp r4,r5 @ actualiza al menor

ble nomenor

primero:mov r4,r5

mov r6,r0

nomenor:

add r1,#4

subs r2,#1

bne otro2

loop: b loop

VECTOR: .word 1000,-1000,234,-234,100,-90,-10,200,-100,-100,20,80,100,200,-400,160

.balign 4

.space(200)

STACK:

.end

**Ejercicio 11**

Dada una cadena de caracteres con terminación nula la cual contiene palabras separadas por espacio, realizar un programa que pase a minúscula todas las letras con excepción de la primera de cada palabra que deberá ser pasada a mayúscula, además, las palabras de menos de 3 letras deberán quedar en minúscula.

ejemplo

entrada

VECT = "El mUndo dE Hoy",0

salida

VECT = "el Mundo de Hoy",0

mov r1,#0

mov r4,#0

mov r2,#VECT

otro: ldrb r3,[r2]

cmp r3,#0

cmpne r3,#' '

beq espacio

cmp r3,#'a'

addlo r3,r3,#('a'-'A')

strb r3,[r2]

add r4,r4,#1

seguir: add r2,r2,#1

b otro

espacio:

cmp r4,#3

blo seguir\_esp

ldrb r5,[r2,-r4]

sub r5,r5,#('a'-'A')

strb r5,[r2,-r4]

seguir\_esp:

mov r4,#0

cmp r3,#0

bne seguir

salir:

loop: b loop

/\* ==============================

\* CONTANTES

\* ==============================

\*/

VECT: .asciz "oraCiOn PAra vER sI fUNCIonA"

.balign 4

.end

**Ejercicio 12**

Se debe realizar un programa que calcule de los 100 primeros números naturales cuales son primos usando el método de la "Criba de Eratóstenes". El método se resume en estos pasos:

* se carga un vector de 99 elementos con los números naturales del 2 hasta 100.
* se comienza por el numero 2, lo dejamos y eliminamos a partir de allí todos los números múltiplo de 2.
* luego nos corremos hasta el próximo número no borrado ( en este caso el 3) y eliminamos todos los múltiplos de 3.
* el siguiente numero sin eliminar será el 5, lo dejamos y eliminamos todos los múltiplos de 5.
* así vamos avanzando, cuando llegamos a un número que no ha sido eliminado lo dejamos y a partir de allí eliminamos los múltiplos de él.

Al final copiamos a un nuevo vector todos los números no eliminados.

mov r1,#VECT

mov r3,#2

mov r7,#VECT + ( 51 - 2) @ verificar un numero mayor a 51 no tiene sentido

@ crear vector de 99 elementos (2 - 100)

mov r2,r1

otro1:

strb r3,[r2],#1

add r3,r3,#1

cmp r3,#101

bne otro1

@ busca elemento no borrado y borra sus multiplos

mov r2,r1

otro2:

ldrb r3,[r2] @ lee de r2 el numero a verificar

cmp r3,#0 @ si es cero busca otro numero

beq seguir

@ si no es cero borra los multiplos

add r4,r3,r2 @ calcula la posicion del primer nro a borrar

mov r5,#0

borrar: strb r5,[r4],r3 @ borra saltando r3 posiciones hasta final de vector

cmp r4,#FINVEC

bls borrar

seguir: add r2,r2,#1

cmp r2,r7 @ si llegó a fin de vector sale ( en realidad se realiza el proceso hasta 51)

bne otro2

salir:

loop: b loop

/\* ==============================

\* CONTANTES

\* ==============================

\*/

VECT: .space 99 ,0

FINVEC:

PROM: .hword 0

.balign 4

.end

**Ejercicio 13**

Se debe realizar un programa que invierta la posición de las letras en cada una de las palabras guardadas en un vector. Las características de este vector a tener en cuenta son:

* Las palabras en el mismo están separadas por uno o varios espacios.
* No existen signos de puntuación.
* El vector termina con nulo.

ejemplo

vec1 = " Las palabras en el mismo están separadas por uno o varios espacios ",0

resultado

vec1 = " saL sarbalap ne le omsim nátse sadarapes rop onu o soirav soicapse ",0

mov r1,#VECT-1

mov r4,#0

mov r2,#VECT

otro: ldrb r3,[r2]

cmp r3,#0

cmpne r3,#' '

beq espacio

add r4,r4,#1

seguir: add r2,r2,#1

b otro

espacio:

cmp r4,#1

bls seguir\_esp

mov r5,#1

otracopia:

ldrb r7,[r1,r5]

ldrb r6,[r2,-r5]

strb r7,[r2,-r5]

strb r6,[r1,r5]

cmp r5,r4,lsr #1

add r5,r5,#1

bne otracopia

seguir\_esp:

mov r1,r2

mov r4,#0

cmp r3,#0

bne seguir

salir:

loop: b loop

/\* ==============================

\* CONTANTES

\* ==============================

\*/

VECT: .asciz " Las palabras en el mismo están separadas por uno o varios espacios "

.balign 4

.end

**Ejercicio 14**

Partiendo de una cadena de caracteres y 10 vectores de 10 punteros cada uno, se pide realizar un programa que guarde en esos vectores las palabras encontradas en la cadena de la siguiente forma: En el primer vector todas las palabras que tengan menos de 3 letras, luego en el segundo las que tengan 4 letras, en el próximo 5, hasta llegar al décimo vector, donde se guardarán aquellas que tengan mas de 11 letras. En caso de poseer mas de 10 palabras de una clase, no se guardarán.

Nota: no es necesario guardar las palabras en otro vector, simplemente con guardar el puntero de comienzo de la palabra y luego cambiar el espacio que separa esta de la próxima por Nulo es suficiente.

reset: ldr sp,=STACK

ldr r3,=VECTORES

ldr r2,=CADENA

otrapalabra:

mov r4,#0

mov r1,r2 @ puntero a la primera letra

otro: ldrb r0,[r2],#1 @ buscar palabra

cmp r0,#' '

beq finpal

cmp r0,#0

beq finpal

add r4,#1 @ contar letras

b otro

finpal:

cmp r4,#0

beq continuar

cmp r4,#3 @ calcula vector a guardar según el largo de la palabra

movlo r4,#3 @ para las menores a 3 letras la deja como 3

cmp r4,#12

movhi r4,#12 @ y para las mayores a 12 las deja como 12

sub r4,#3 @ restando 3 al resultado queda todo comprendido entre 0 y 9 donde este valor indica el vector a usar

mov r5,#40

mul r4,r5 @ se multiplica por 40 para calcular la posicion dentro del vector general

mov r5,#10

otrolugar:

ldr r6,[r3,r4] @ busco un lugar vacio

cmp r6,#0

beq grabar

add r4,#4

subs r5,#1

bne otrolugar

b continuar

grabar:

str r1,[r3,r4] @ se graba el puntero de la primer letra en el lugar vacio

mov r5,#0

strb r5,[r2,#-1] @ y un nulo en el espacio siguiente a la última letra

continuar:

cmp r0,#0

bne otrapalabra

loop: b loop

CADENA: .ascii "hola es la mundo palabra varios"

.balign 4

@ se utiliza un solo vector de 10 \* 10, formado por 10 filas de 10 elementos, cada fila representa un vector

VECTORES:

.rept 10

.word 0,0,0,0,0,0,0,0,0,0

.endr

.space(200)

STACK:

.end

**Ejercicio 15**

Dada una cadena de bytes sin signo de terminación nula, por cada elemento de esta cadena se sacará el promedio del mismo con los tres bytes anteriores guardando este valor en una segunda cadena en la posición correspondiente. La excepción a esta regla son los tres primeros valores que son guardados directamente en la segunda cadena sin calculo de promedio.

Ejemplo

cadena1 = 2,14,7,8,9,2,.......,0

salida

cadena2 = 2,14,7,7,9,6

donde 2 <- 2, 14 <- 14, 7 <- 7, 7 <- prom(2+14+7+8), 9 <- prom(14+7+8+9), 6 <- prom(7+8+9+2)

reset:

mov r1,#0

mov r2,#VECT

mov r3,#0

otro: mov r2,#VECT

ldrb r4,[r2,r1]

cmp r4,#0

beq loop

add r3,r3,r4

cmp r1,#3

blo seguir

sub r5,r1,#4

ldrb r4,[r2,r5]

subne r3,r3,r4

mov r4,r3, asr #2

seguir: mov r2,#GUARDAR

strb r4,[r2,r1]

add r1,r1,#1

b otro

loop: b loop

/\* ==============================

\* CONTANTES

\* ==============================

\*/

VECT: .byte 10,20,30,40,50,60,70,80,90,100,110,120,130,140,-10,-20,0

GUARDAR:.space 30

.balign 4

.end

**Ejercicio 16**

Una cadena de caracteres debe se encriptada, para ello se utilizará un método muy básico, el cual consiste en sumar al código ascii de cada letra en la cadena a encriptar el código ascii del caracter correspondiente de una clave, cada vez que se alcanza el final de esta, se vuelve a comenzar con su primer caracter.

Ejemplo

aencrip = 'h','o','l','a',' ','m','u','n','d','o',0

clave = 'f','1','2',0

resultado = 'h'+'f' , 'o'+'1' , 'l'+'2' , 'a'+'f' , ' '+'1' , 'm'+'2' , 'u'+'f' , 'n'+'1' , 'd'+'2' , 'o'+'f'

reset:

mov r2,#VECT

mov r3,#CLAVE

mov r1,r3

@ lee un byte del vector

otro:

ldrb r4,[r2]

cmp r4,#0

beq salir

@ lee un byte de la clave

otra\_clave:

ldrb r5,[r1],#1

cmp r5,#0

moveq r1,r3 @ si el byte leido es 0, comienza nuevamente a leer

beq otra\_clave @ el primer byte

@ suma los dos valores y guarda el resultado en el vector

add r4,r4,r5

strb r4,[r2]

add r2,r2,#1

b otro

salir:

loop: b loop

/\* ==============================

\* CONTANTES

\* ==============================

\*/

VECT: .asciz "cadena a encriptar"

CLAVE: .asciz "afg2"

.balign 4

.end

**Ejercicio 17**

En un vector de caracteres, el cual posee solo palabras separadas por espacios, se debe buscar la palabra mas larga, luego, formar otro vector con las mismas palabras con el agregado a cada una de n caracteres '\_' donde n es la diferencia entre la palabra mas larga y la palabra actual.

ejemplo

entrada = "igualar el largo de las palabras",0

salida = "igualar\_ el\_\_\_\_\_\_ largo\_\_\_ de\_\_\_\_\_\_ las\_\_\_\_\_ palabras",0

reset:

ldr r1,=CADENA

ldr r2,=SALIDA

mov r5,#0 @maxima longitud

mov r4,#0 @contador longitud

otro: ldrb r3,[r1],#1

cmp r3,#0

beq salir1

cmp r3,#' '

beq salir1

add r4,#1

b otro

salir1: cmp r4,r5

movhi r5,r4

mov r4,#0

cmp r3,#0

bne otro

ldr r1,=CADENA

ldr r2,=SALIDA

otro2: ldrb r3,[r1],#1

cmp r3,#0

beq salir2

cmp r3,#' '

beq salir2

strb r3,[r2],#1

add r4,#1

b otro2

salir2: cmp r4,#0 @si es otro espacio seguido no completo con '\_'

beq seguir

otro3: cmp r4,r5

beq seguir

mov r6,#'\_'

strb r6,[r2],#1

add r4,#1

b otro3

seguir: strb r3,[r2],#1

mov r4,#0

cmp r3,#0

bne otro2

loop: b loop

.ltorg

/\* ==============================

\* CONTANTES

\* ==============================

\*/

CADENA: .asciz "igualar el largo de las palabras"

SALIDA: .space(200)

.balign 4

.end

**Ejercicio 18**

Realizar un programa que elimine de un vector de byte sin signo previamente cargado, los números múltiplos de 4.

El vector se denomina vec1 y termina con 0.

ejemplo

entrada en vec1 = 80,5,6,18,4,5,20,19,0

salida en vec1 = 5,6,18,5,19,0

archivo fuente ex1.s

**Ejercicio 19**

Dado dos vectores previamente cargado de caracteres alfanuméricos (letras mayúsculas y minúsculas del alfabeto ingles, espacios y números) y terminación nula, desarrollar un programa que copie del vector 1 a un nuevo vector 3 donde se insertará por cada caracter que coincida con el primer caracter del vector 2 (COMPARACIÓN CASE INSENSITIVE) el propio vector 2 (sin su primer caracter).

Ejemplo

Entrada

vector1 = "insertar en 1 la palabra 1"

vector2 = "1mundo"

Salida

vector3 = "insertar en mundo la palabra mundo"

reset: mov r1, #VECT1

mov r2, #VECT2

add r3, r2,#1

mov r4, #SALIDA

@ lee la primera letra de VECT2 y la pasa a mayuscula

ldrb r5,[r2]

cmp r5,#'a'

subhs r5, r5, #('a'-'A')

@ bucle principal

otro: ldrb r6, [r1],#1

cmp r6, #0

beq salir @ sale al finalizar barrido de VECT1

mov r7,r6 @ copia en r7 el valor leido de VECT1 para no afectarlo con el pase a mayuscula

cmp r6, #'a' @ pasa a mayuscula el valor veido

subhs r6, r6, #('a'-'A')

cmp r5, r6 @ compara el valor leido con el primer elemento de VECT2

strneb r7, [r4], #1 @ si no son iguales guarda el valor leido de VECT1 en SALIDA

bleq insertar @ si eran iguales llama a insertar

b otro

/\* ==================================================================================

\* función insertar

\* copia desde el 2do elemento de VECT2 hasta su final en SALIDA

\* ==================================================================================\*/

insertar:

mov r2,r3 @ guarda en r2 el puntero al segundo elemento de VECT2

insertarotra:

ldrb r6, [r2],#1

cmp r6, #0

moveq pc,lr @ en esta linea retorno desde donde fué llamada la funcion (si r6 = 0)

strb r6, [r4], #1

b insertarotra

@ fin del programa

salir: strb r6,[r4],#1 @ copio el null al final de la cadena

loop: b loop

/\* ==============================

\* CONSTANTES

\* ==============================

\*/

VECT1: .asciz "insertar en X la palabra x"

VECT2: .asciz "xmundo"

SALIDA: .space 50

.balign 4

.end

**Ejercicio 20**

Dado dos vectores previamente cargados de caracteres alfanuméricos (letras mayúsculas y minúsculas del alfabeto ingles, espacios y números) y terminación nula, desarrollar un programa que cuente la cantidad de veces que el vector 2 se repite en el vector 1.

Ejemplo

Entrada

vector1 = "esto es una prueba de palabras repetidas"

vector2 = "es"

Salida

cantidad = 2

reset:

mov r1,#VECT1

mov r2,#VECT2

mov r9,#0 @ contador

/\* bucle principal

\* se recorre VEC1 hasta el final y por cada lectura, llama a la funcion comparar

\*/

otro:

ldrb r4,[r1],#1

cmp r4,#0

beq salir

bl comparar

b otro

/\* =========================================================

\* funcion comparar

\* compara las cadenas VEC1(comenzando desde la posicion

\* apuntada por r1-1) y VEC2 hasta el final de VEC2

\* =========================================================\*/

comparar:

sub r5,r1,#1 @ utiliza como puntero inicial de VEC1 la posicion anterior a r1

mov r6,r2 @ el puntero de VEC2 apunta al comienzo del mismo

otracmp:

ldrb r8,[r6],#1 @ lee VEC2

cmp r8,#0 @ si VEC2 llega al final, incrementa r9 (son iguales ) y sale

addeq r9,r9,#1

beq retorno

ldrb r7,[r5],#1 @ lee VEC1

cmp r7,r8

beq otracmp @ si son iguales compara siguiente, si son distintos sale

retorno:

mov pc,lr

salir:

loop: b loop

/\* ==============================

\* CONTANTES

\* ==============================

\*/

VECT1: .asciz "esto es una prueba de palabras"

VECT2: .asciz "es"

.balign 4

.end

**Ejercicio 21**

Dado un vector previamente cargado de caracteres alfanuméricos (letras mayúsculas y minúsculas del alfabeto ingles, espacios y números) y terminación nula, desarrollar un programa que por cada carácter verifique cuantos caracteres a continuación están repetidos, se dispone para guardar los datos de dos vectores, uno donde se dispone el carácter (tipo byte) y en otro la cantidad de repeticiones de ese carácter (tipo halfword).

Ejemplo

Entrada

vector = "fssdffjlllleiiiilv4"

Salida

letras = 'f','s','d','f','j','l','e','i','l','v','4'

cantidad = 1 , 2 , 1 , 2 , 1 , 4 , 1 , 4 , 1 , 1 , 1

reset:

mov r1,#VECT @ vector origen

mov r2,#VEC1 @ vector de byte destino con las letras

mov r3,#VEC2 @ vector de halfword destino con las cantidades de cada letra

mov r6,#0 @ ultima letra guardada (inicia con 0 para que la comparacion sea negativa y entre a agregar)

mov r7,#1 @ contador de letras repetidas (inicia con 1 sin embargo la primera pasada si o si entra a agregar

@ donde se vuelve a inicializar con 1, esta linea entonces no es necesaria)

otro1: ldrb r5,[r1],#1 @ lee la letra de destino

cmp r5,#0 @ si es 0 llego a fin de cadena y sale

beq final

cmp r5,r6 @ comparar con la ultima letra guardada ( la primera pasada r6 es 0 entonces esta comp da diferente si o si)

bne agregar @ si son distintas voy a agregar

add r7,r7,#1 @ si las letras eran iguales incremento r7 y lo guardo en r3 - 2

strh r7,[r3,#-2]

continuar:

b otro1

/\* ==========================================================================

\* agregar: agrega una nueva entrada a los dos vectores destino, guardando en

\* VEC1 la letra leida y en VEC2 inicializando con 1

\* si bien se plantea como funcion, no lo es, la devolucion de una función se

\* debe realizar utilizando el lr no con una instruccion b.

\* esta funcion guarda la letra e inicializa con 1 el vector VECT2 con una

\* instruccion de postincremento, esto significa que cuando se quiera acceder a

\* la cuenta de la ultima letra guardada, se debe restar 2 al valor de r3

\* ==========================================================================\*/

agregar:

mov r7,#1

strb r5,[r2],#1 @ guarda la ultima letra leida

strh r7,[r3],#2 @ inicializa cuenta con 1

mov r6,r5 @ actualiza r6 con la nueva letra

b continuar

final:

loop: b loop

/\* ==============================

\* CONTANTES

\* ==============================

\*/

VECT: .asciz "fssddgggrrteeeeeeeew"

VEC1: .space(21)

VEC2: .space(21)

.end

**Ejercicio 22**

Realizar una función que reciba como parámetro el puntero a una cadena de terminación nula, la misma deberá transformar la cadena recibida a un número.

Finalmente se deberá utilizar la función en un bucle donde: se tome un valor de un vector que posee el puntero a la cadena (se supone ya cargado), se envíe a la función y el resultado se guarde en otro vector (no es necesario inicializarlo), la acción se repite hasta concluir los 16 ciclos de conversión.

IMPORTANTE:

* El número en ASCII que se encuentra en la cadena es base 16.
* La cadena recibida se supone que solo posee dígitos hexadecimales -donde las letras se encuentran en mayúsculas-, puntos de separación de miles que deben ser descartados o un signo.
* La cadena nunca contendrá un número mayor a 32 bits.
* El único signo posible es un '-' y como primer byte, indicando que el numero es negativo, si este signo no existe el número es positivo.

Version 1

reset: ldr sp,=STACK

ldr r1,TAMANIO

ldr r2,=VECTOR

ldr r3,=NUMEROS

otro: ldr r0,[r2],#4

bl anumero

str r0,[r3],#4

subs r1,#1

bne otro

loop: b loop

/\* Funcion de pase a numero

\* Transforma una cadena en ascii a un numero.

\* Esta cadena posee las siguientes características:

\* \* El número en la cadena se encuentra en base 16

\* \* La cadena se supone que solo posee dígitos hexadecimales, donde las letras se encuentran en mayúsculas, puntos de

\* \* El único signo posible es un '-' y como primer byte, indicando que el número es negativo, si este signo no existe el número es positivo.

\* Entrada

\* Dirección de la cadena en r0

\* Salida

\* Numero resultante en r0

\*/

anumero:

push {r1-r4}

mov r4,#0

ldrb r1,[r0] @ si el primer caracter es '-' se activa la bandera de signo a 1 (r4)

cmp r1,#'-'

moveq r4,#1

addeq r0,#1 @ si es negativo desplazo el puntero para no encontrar de nuevo el signo

mov r2,#0

mov r3,#16

anumero\_otro:

ldrb r1,[r0],#1 @ carga los elementos desde el mas significativo (primer caracter

cmp r1,#0

beq anumero\_fin

cmp r1,#'.'

beq anumero\_cont

cmp r1,#'9' @ si es menos a '9' es un número si es mayor es una letra

subls r1,#'0'

subhi r1,#('A'-10)

mul r2,r3 @ multiplico acumulador por 16

add r2,r1 @ sumo al acumulador el digito

@ ejemplo

@ entrada = '3','4','A'

@ calculo = ((0 \* 16 + 3 ) \* 16 + 4) \* 16 + A

anumero\_cont:

b anumero\_otro

anumero\_fin:

mov r0,r2

cmp r4,#1 @ verifico signo para negar el número si fuera negativo

rsbeq r0,r0,#0

pop {r1-r4}

mov pc,lr

VECTOR: .word NUM1,NUM2

NUMEROS: .word 0,0

TAMANIO: .word 2

NUM1: .asciz "0.A334"

NUM2: .asciz "-98"

.balign 4

.space(200)

STACK:

.end

Version 2

reset: ldr sp,=STACK

ldr r1,TAMANIO

ldr r2,=VECTOR

ldr r3,=NUMEROS

otro: ldr r0,[r2],#4 @ bucle que carga el puntero a la cadena

bl anumero @ pasa cadena a numero

str r0,[r3],#4 @ guarda el resultado

subs r1,#1

bne otro

loop: b loop

/\* Funcion de pase a numero

\* Transforma una cadena en ascii a un numero.

\* Esta cadena posee las siguientes características:

\* \* El número en la cadena se encuentra en base 16

\* \* La cadena se supone que solo posee dígitos hexadecimales, donde las letras se encuentran en mayúsculas, puntos de

\* \* El único signo posible es un '-' y como primer byte, indicando que el número es negativo, si este signo no existe el número es positivo.

\* Entrada

\* Dirección de la cadena en r0

\* Salida

\* Numero resultante en r0

\*/

anumero:

push {r1-r6}

mov r4,#0

ldrb r1,[r0] @ si el primer caracter es '-' se activa la bandera de signo a 1 (r4)

cmp r1,#'-'

moveq r4,#1

addeq r0,#1 @ si es negativo desplazo el puntero para no encontrar de nuevo el signo

sub r5,r0,#1 @ calcular r5 para detectar comienzo de cadena

anumero\_otro:

ldrb r1,[r0],#1 @ bucle para encontrar el final de cadena

cmp r1,#0

bne anumero\_otro

sub r0,#2

mov r2,#1

mov r3,#16

mov r6,#0

anumero\_otro2:

ldrb r1,[r0],#-1 @ carga los elementos desde el menos significativo (último caracter)

cmp r0,r5 @ compara si es comienzo de cadena para salir

blo anumero\_fin

cmp r1,#'.'

beq anumero\_cont

cmp r1,#'9' @ si es menos a '9' es un número si es mayor es una letra

subls r1,#'0'

subhi r1,#('A'-10)

mul r1,r2,r1 @ multiplico el dígito con el peso de la posición

mul r2,r3,r2 @ recalculo el peso de la posición

add r6,r1 @ acumulador con resultado

@ ejemplo

@ entrada = '3','4','A'

@ calculo = (A \* 1)+ (4 \* 1 \* 16) + (3 \* 1 \* 16 \* 16)

anumero\_cont:

b anumero\_otro2

anumero\_fin:

mov r0,r6

cmp r4,#1 @ verifico signo para negar el número si fuera negativo

rsbeq r0,r0,#0

pop {r1-r6}

mov pc,lr

VECTOR: .word NUM1,NUM2

NUMEROS: .word 0,0

TAMANIO: .word 2

NUM1: .asciz "0.A334"

NUM2: .asciz "-98"

.balign 4

.space(200)

STACK:

.end

**Ejercicio 23**

Partiendo de un vector previamente cargado con una cadena alfanumérica de terminación nula, se debe construir un programa que encuentre los números (base 16) contenidos en la cadena, estos números serán cargados en dos vectores de la siguiente forma: en el primero la posición del primer dígito dentro de la cadena (o su puntero), en el segundo vector la longitud de dicho número, esto se repetirá hasta terminar de barrer la cadena.

IMPORTANTE:

* Las características de los números a encontrar serán las siguientes: comienzan con un dígito decimal (0 a 9) o el signo menos (-), pueden contener separaciones de miles (.) o coma decimal (,)
* Se supone los vectores destino ya inicializados y con capacidad suficiente (ambas de tipo word).

Ejemplo

entrada = "hola 98A,Bmundo-0A.334"

posicion = 5,15

largo = 5,7

reset: ldr sp,=STACK

ldr r1,=VECTOR

ldr r2,=POSICION

ldr r3,=TAMANIO

otro: ldrb r0,[r1],#1

cmp r0,#0

beq salir

cmp r0,#'-'

beq pasar

bl esnumero

cmp r0,#0

beq continuar @ si no es signo menos o dígito, continua

pasar:

/\* contar el resto del número hasta encontra un no dígito y no signo \*/

sub r0,r1,#1

str r0,[r2],#4 @ guarda la posición de signo o dígito encontrado

mov r4,#1 @ contador de dígitos

otro\_d: ldrb r0,[r1],#1

cmp r0,#0

beq salir\_num

cmp r0,#'.' @ verificar si es punto

beq contar

cmp r0,#',' @ verificar si es coma

beq contar

mov r5,r0

bl esnumero @ verificar si es numero

cmp r0,#1

beq contar

mov r0,r5

bl esletra @ verificar si es letra

cmp r0,#1

beq contar

b salir\_num

contar:

add r4,#1

b otro\_d

salir\_num:

str r4,[r3],#4 @ cuarda el conteo

continuar:

b otro

salir:

loop: b loop

/\*

devuelve si un ascii es un digito decimal

entrada r0 con el ascii

salida r0=0 no es digito r0=1 es digito

\*/

esnumero:

cmp r0,#'0'

blo esnumero\_no

cmp r0,#'9'

bhi esnumero\_no

mov r0,#1

b esnumero\_salir

esnumero\_no:

mov r0,#0

esnumero\_salir:

mov pc,lr

/\*

devuelve si un ascii es una letra hexa (A-F)

entrada r0 con el ascii

salida r0=0 no es letra r0=1 es letra

\*/

esletra:

cmp r0,#'A'

blo esletra\_no

cmp r0,#'F'

bhi esletra\_no

mov r0,#1

b esletra\_salir

esletra\_no:

mov r0,#0

esletra\_salir:

mov pc,lr

VECTOR: .asciz "hola3442,323 mundo0A.3345"

.balign 4

POSICION: .word 0,0,0,0,0,0,0,0,0,0,0,0

TAMANIO: .word 0,0,0,0,0,0,0,0,0,0,0,0

.space(200)

STACK:

.end