Programas en ensamblador ARM64 para macOS: de un ejemplo sencillo a casos más complejos

1) Programa ARM64 en ensamblador para mostrar “Hola mundo” en macOS

Este es el más básico: carga una cadena y llama a puts.

```
.section __TEXT,__text
.globl _main
.p2align 2_main:
stp x29, x30, [sp, #-16]!
mov x29, sp    adrp x0, mensaje@PAGE
add  x0, x0, mensaje@PAGEOFF
bl   _puts    mov w0, #0
ldp x29, x30, [sp], #16
ret.section __TEXT,__cstring
mensaje:
.asciz "Hola mundo desde ARM64 en macOS"
```

Compilación

```
clang hola.s -o hola
./hola
```

---

2) Programa ARM64 en ensamblador para sumar dos números e imprimir

el resultado en macOS

Aquí ya usas registros, haces una suma y llamas a printf.

```
.section __TEXT,__text
.globl _main
.p2align 2_main:
stp x29, x30, [sp, #-16]!
mov x29, sp    mov x1, #7
mov x2, #5
add x3, x1, x2    adrp x0, formato@PAGE
add  x0, x0, formato@PAGEOFF
bl   _printf    mov w0, #0
ldp x29, x30, [sp], #16
ret.section __TEXT,__cstring
formato:
.asciz "7 + 5 = %ld\n"
```

Compilación

```
clang suma.s -o suma
./suma
```

---

3) Programa ARM64 en ensamblador para calcular la longitud de una cadena en macOS

Aquí ya recorres memoria byte a byte, como una strlen manual.

```
.section __TEXT,__text
.globl _main
.p2align 2_main:
stp x29, x30, [sp, #-16]!
mov x29, sp    adrp x1, texto@PAGE
add  x1, x1, texto@PAGEOFF    mov x2, #0              //
contadorbucle:
ldrb w3, [x1, x2]
```

```
cbz  w3, fin_bucle
add  x2, x2, #1
b    buclefin_bucle:
adrp x0, formato@PAGE
add  x0, x0, formato@PAGEOFF
mov  x1, x2
bl   _printf    mov w0, #0
ldp x29, x30, [sp], #16
ret.section __TEXT,__cstring
texto:
.asciz "Apple Silicon"
formato:
.asciz "Longitud: %ld\n"
```

Compilación

```
clang strlen.s -o strlen
./strlen
```

---

4) Programa ARM64 en ensamblador para contar vocales en una cadena en macOS

Este ya tiene más lógica: bucle, comparaciones y recuento.

```
.section __TEXT,__text
.globl _main
.p2align 2_main:
stp x29, x30, [sp, #-16]!
mov x29, sp    adrp x1, texto@PAGE
add  x1, x1, texto@PAGEOFF    mov x2, #0      // índice
mov x4, #0      // contador de vocalesbucle:
ldrb w3, [x1, x2]
cbz  w3, fin    cmp  w3, #'a'
b.eq es_vocal
cmp  w3, #'e'
b.eq es_vocal
cmp  w3, #'i'
```

```
b.eq es_vocal
cmp  w3, #'o'
b.eq es_vocal
cmp  w3, #'u'
b.eq es_vocal    cmp  w3, #'A'
b.eq es_vocal
cmp  w3, #'E'
b.eq es_vocal
cmp  w3, #'I'
b.eq es_vocal
cmp  w3, #'O'
b.eq es_vocal
cmp  w3, #'U'
b.eq es_vocalsiguiente:
add x2, x2, #1
b   buclees_vocal:
add x4, x4, #1
b   siguientefin:
adrp x0, formato@PAGE
add  x0, x0, formato@PAGEOFF
mov  x1, x4
bl   _printf    mov w0, #0
ldp x29, x30, [sp], #16
ret.section __TEXT,__cstring
texto:
.asciz "Automatizacion normativa con ARM64"
formato:
.asciz "Numero de vocales: %ld\n"
```

Compilación

```
clang vocales.s -o vocales
./vocales
```

---

5) Programa ARM64 en ensamblador

para abrir un fichero y mostrar su contenido en macOS

Este ya sube bastante el nivel: llama a open, read, write y close usando libc.

Lee el fichero entrada.txt y lo imprime por pantalla.

```
.section __TEXT,__text
.globl _main
.p2align 2_main:
stp x29, x30, [sp, #-32]!
mov x29, sp    // open("entrada.txt", 0)
adrp x0, nombre_archivo@PAGE
add  x0, x0, nombre_archivo@PAGEOFF
mov  x1, #0
bl   _open    // guardar fd
mov x19, x0    // si fd < 0, error
cmp x19, #0
b.lt error    // read(fd, buffer, 255)
mov  x0, x19
adrp x1, buffer@PAGE
add  x1, x1, buffer@PAGEOFF
mov  x2, #255
bl   _read    // x0 = bytes leídos
mov x20, x0    // write(1, buffer, bytes_leidos)
mov  x0, #1
adrp x1, buffer@PAGE
add  x1, x1, buffer@PAGEOFF
mov  x2, x20
bl   _write    // close(fd)
mov x0, x19
bl  _close    mov w0, #0
ldp x29, x30, [sp], #32
reterror:
adrp x0, msg_error@PAGE
add  x0, x0, msg_error@PAGEOFF
bl   _puts    mov w0, #1
ldp x29, x30, [sp], #32
ret.section __DATA,__data
```

```
buffer:
.space 256.section __TEXT,__cstring
nombre_archivo:
.asciz "entrada.txt"
msg_error:
.asciz "No se pudo abrir el fichero"
```

Compilación

```
clang fichero.s -o fichero
./fichero
```

Crea antes el fichero de prueba

```
echo "Hola, este texto viene de un fichero." > entrada.txt
```