04 Lenguaje ensamblador, svc y transferencia



Arquitectura de Computadoras y Ensambladores 1
M.Sc. Luis Fernando Espino Barrios

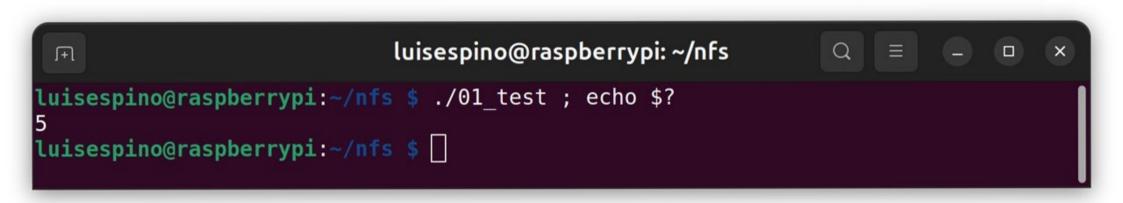
Directivas

- Declaración de símbolos globales: .global o .globl
- Declaración de código en el segmento text: .text
- Declaración de datos en el semento data: .data
- Declaración de cadenas: .ascii o .asciz

Etiquetas

- Las etiquetas son nombres de una ubicación específica del código.
- En la llamada solo se escribe el nombre y en la ubicación se agregan (:) dos puntos al final.
- Entre algunas ubicaciones importantes están:
 - Inicio del código: _start
 - Puntos de referencia
 - Inicio o fin de ciclos
 - Destino de saltos condicionales e incondicionales.

```
1 .global _start
2
3 _start:
4 mov x0, 5 // return value
5 mov x8, 93 // exit syscall_num
6 svc 0 // generic syscall
```



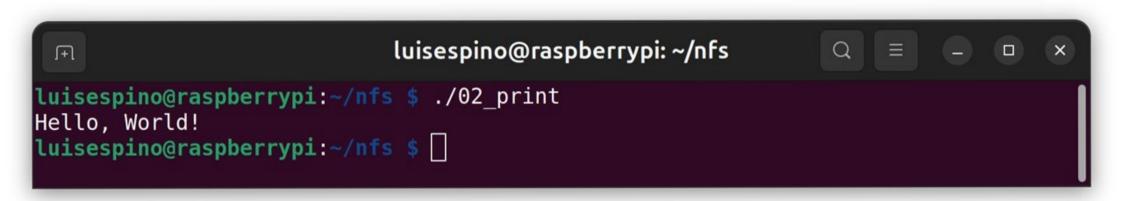
Carga y almacenamiento de registro simple

Instrucción	Descripción
ldr Rd, <addr></addr>	Carga un registro de la memoria (puede usar una dirección, un offset, una etiqueta, un inmediato o un símbolo)
str Rd, <addr></addr>	Almacena un registro en la memoria (puede usar una dirección, un offset, una etiqueta, un inmediato o un símbolo)

Operaciones de movimiento de datos

Instrucción	Descripción
mov Rd, Rn	Mueve (copia) un registro Rn a otro Rd (puede usar un registro Rn, un inmediato o un patrón)
movz Rd, imm{, lsl <shift>}</shift>	Establece un inmediato con corrimiento y llenado de ceros hacia un registro Rd.
movn Rd, imm{, Isl <shift>}</shift>	Igual que movz solo que negado o complemento a 1.
movk Rd, imm{, lsl <shift>}</shift>	Establece un inmediato con corrimiento hacia un registro Rd, dejando el resto igual.

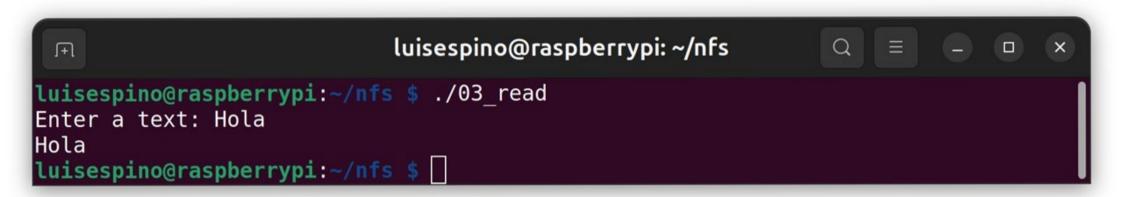
```
ASM 02 print.s
     .global start
 3
     start:
 4
         mov \times 0, 1 // set stdout
 5
         ldr x1, =msg // load msg
 6
      mov \times 2, 14 // size msg
         mov x8, 64 // write syscall num
 8
                     // generic syscall
         svc 0
 9
10
         mov \times 0, 0 // return value
11
         mov x8, 93 // exit syscall num
         svc 0
12
                     // generic syscall
13
14
     .data
15
     msg: .asciz "Hello, World!\n"
```



Otras directivas

- Declaración de una sección específica: .section
- Declaración de constantes simbólicas: .equ
- Declaración de datos no inicializados: .bss
- Declaración de espacio para almacenar: .space

```
03_read.s
      .global start
      start:
          mov \times 0, 1
                           // set stdout
          ldr x1, =msg
                          // load msg
          mov x2, 18
                          // size msg
          mov x8, 64
                           // write syscall num
          svc 0
                           // generic syscall
          mov \times 0, 0
                          // set stdin
11
          ldr x1, =buf
                          // load buffer
12
          mov x2, 16
                          // size buffer
13
          mov x8, 63
          svc 0
                           // generic syscall
15
          mov \times 0, 1
                          // set stdout
          ldr x1, =buf
                          // load buffer
          mov x8, 64
                          // write syscall num
          svc 0
                           // generic syscall
21
          mov \times 0, 0
                          // return value
22
         mov x8, 93
                          // exit syscal num
23
          SVC 0
                          // generic syscall
24
25
     .data
     msg:
          .ascii "Enter a text: "
29
     .bss
     buf:
          .space 16
```



Ejercicios

- Realizar el proceso de copiado de código, ensamblado, enlazado y ejecución de los tres ejemplos vistos.
- Puede realizarse en Qemu, en UserLAnd o físicamente en la Raspberry.

Bibliografía

- Elahi, A. (2022). Fundamentals of Computer Architecture and ARM Assembly Language. Springer.
- Harris, S. & Harris, D. (2016). Digital Design and Computer Architecture: ARM Edition. Elsiever Inc.
- Mazidi, M. & Otros. (2013). ARM Assembly Language: Programming & Architecture.
- Smith, S. (2019). Raspberry Pi Assembly Language Programming: ARM Processor Coding. Apress.