08 Ramificaciones

Arquitectura de Computadoras y Ensambladores 1
M.Sc. Luis Fernando Espino Barrios
2024

Ramificaciones

- Son saltos en el código, alterando el flujo secuencial.
- En Arm son conocidos como branch.
- Existen branch incondicionales y

- Tanto las instrucciones condicionales como incondicionales están relacionadas con la ramificación.
- Una ramificación es un cambio del flujo de ejecución forzando al Program Counter (PC) apuntar a otra dirección.
- La ramificación incondicional es la que no verifica ninguna condición y solamente cambio el flujo.

Name	Effect	Description
b	pc ← target_address	Unconditionally move new address to the program counter (pc).

b<cond>	**if** <cond> **then**
	  pc ← target_address
	end if
		Conditionally move new address to the program counter (pc).

Branch incondiconal hacia adelante y atrás

```
...
        b forward
...
forward:
...
```

```
...
backward:
..
        b backward
...
```

If-cond-statements-endif

- Es la estructura condicional más básica.
- Se compone de una condición, normalmente negada para no usar doble branch y un bloque de sentencias que se ejecutan despues del branch.
- La condición debe estar acompañada por instrucciones que modifiquen el PSTATE.
- Para comodidad se pueden utilizar etiquetas para identificar el inicio con if: y el fin con endif:.

```
.global _start

.data
msg: .ascii "The number is natural\n"

.text
_start:
    mov x0, 15      // number to test

if:
    cmp x0, 0
    blt endif       // b to endif if x0<0

    mov x0, 1       // print msg
    ldr x1, =msg
    mov x2, 22
    mov x8, 64
    svc 0

endif:
    mov x0, 0       // exit
    mov x8, 93
    svc 0
```

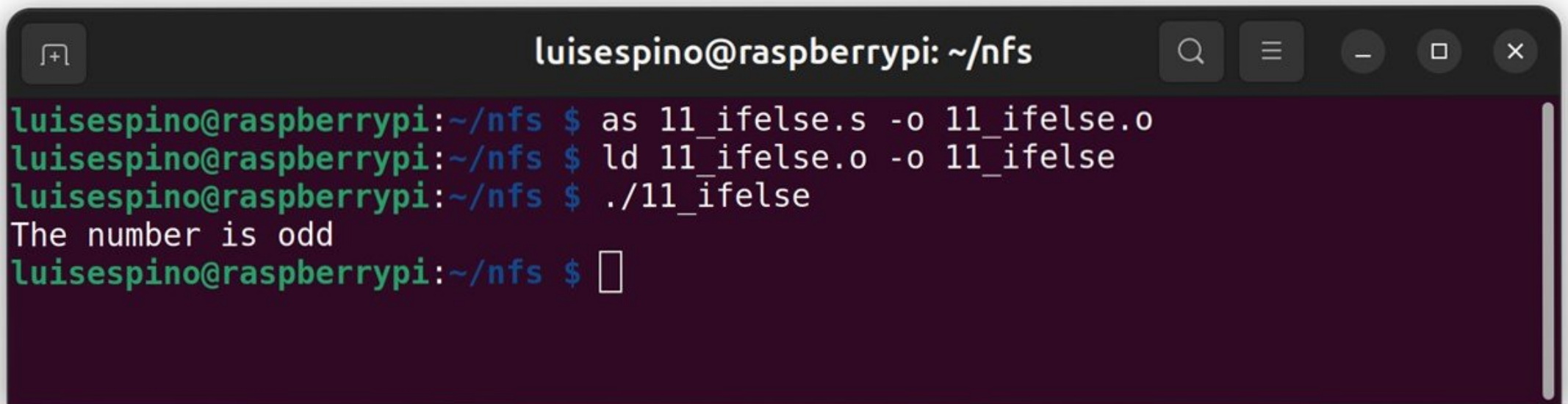

luisespino@raspberrypi: ~/nfs
luisespino@raspberrypi:~/nfs $ as 10_if.s -o 10_if.o
luisespino@raspberrypi:~/nfs $ ld 10_if.o -o 10_if
luisespino@raspberrypi:~/nfs $ ./10_if
The number is natural
luisespino@raspberrypi:~/nfs $

If-cond-statements-else-statements-endif

- Es la estructura que tiene una evaluación mutuamente excluyente de dos valores falso y verdadero.
- Se compone de una condición, normalmente negada para no usar doble branch y dos bloques de sentencias que se ejecutan una despues del branch y otro con la etiqueta de falso.
- La condición debe estar acompañada por instrucciones que modifiquen el PSTATE.
- Para comodidad se pueden utilizar etiquetas para identificar el inicio con if:, el segundo bloque con else: y el fin con endif:.

```
.global _start

.data
odd: .ascii "The number is odd\n"
even: .ascii "The number is even\n"

.text
_start:
    mov x0, 15      // number to test
```

```
if:
    ands x0, x0, 1
    beq else        // b to else if even

    mov x0, 1       // print odd msg
    ldr x1, =odd
    mov x2, 18
    mov x8, 64
    svc 0
    b endif

else:
    mov x0, 1       // print even msg
    ldr x1, =even
    mov x2, 19
    mov x8, 64
    svc 0

endif:
    mov x0, 0       // exit
    mov x8, 93
    svc 0
```

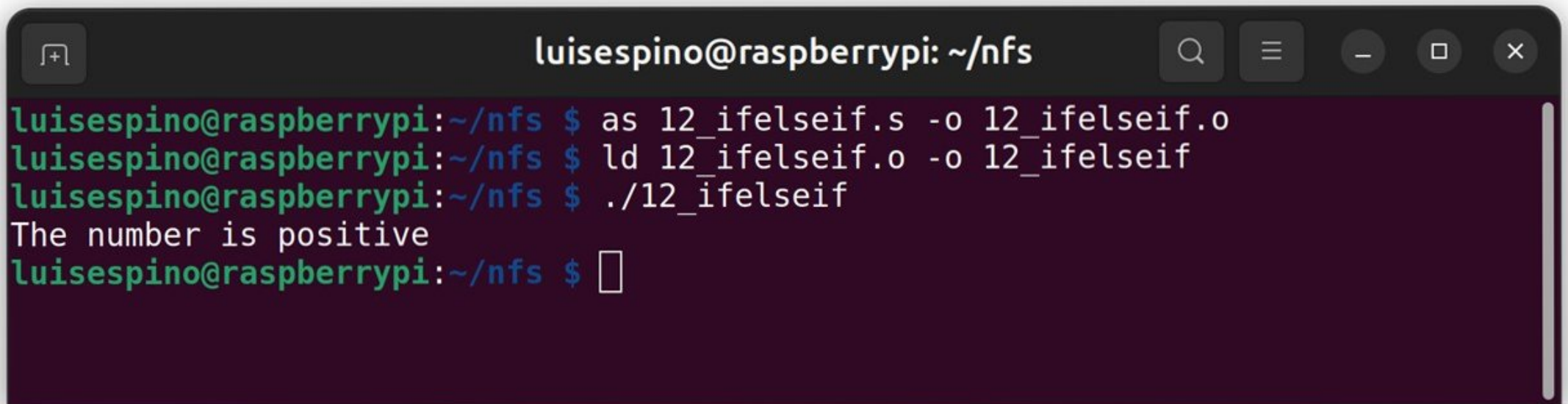

luisespino@raspberrypi: ~/nfs
luisespino@raspberrypi:~/nfs $ as 11_ifelse.s -o 11_ifelse.o
luisespino@raspberrypi:~/nfs $ ld 11_ifelse.o -o 11_ifelse
luisespino@raspberrypi:~/nfs $ ./11_ifelse
The number is odd
luisespino@raspberrypi:~/nfs $

If-cond-stms-elseif-stms-else-stms-endif

- Es la estructura que tiene mas de dos bloques de sentencias.
- Se compone de una condición, normalmente negada para no usar doble branch de falso y verdadero, y tres o más bloques de sentencias que se ejecutan una despues del branch y los otros con las etiqueta necesarias.
- La condición debe estar acompañada por instrucciones que modifiquen el PSTATE.
- Para comodidad se pueden utilizar etiquetas para identificar el inicio con if:, los n bloques con elseif, el último bloque con else: y el fin con endif:.

```
.global _start

.data
pos: .ascii "The number is positive\n"
neg: .ascii "The number is negative\n"
neu: .ascii "The number is neutral\n"

.text
_start:
    mov x0, 15      // number to test

if:
    cmp x0, 0
    blt elseif      // b to elseif if neg
    beq else        // b to else if neutral

    mov x0, 1       // print positive msg
    ldr x1, =pos
    mov x2, 23
    mov x8, 64
    svc 0
    b endif
```

```
elseif:
    mov x0, 1       // print negative msg
    ldr x1, =neg
    mov x2, 23
    mov x8, 64
    svc 0
    b endif

else:
    mov x0, 1       // print nneutral msg
    ldr x1, =neu
    mov x2, 22
    mov x8, 64
    svc 0

endif:
    mov x0, 0       // exit
    mov x8, 93
    svc 0
```


luisespino@raspberrypi: ~/nfs
luisespino@raspberrypi:~/nfs $ as 12_ifelseif.s -o 12_ifelseif.o
luisespino@raspberrypi:~/nfs $ ld 12_ifelseif.o -o 12_ifelseif
luisespino@raspberrypi:~/nfs $ ./12_ifelseif
The number is positive
luisespino@raspberrypi:~/nfs $

Comparar y ramificar

- Existen cuatro instrucciones que comparan si un valor es cero o no cero, y también una comparación con un bit específico. Al mismo tiempo hacen el branch.

cbz	Compare and Branch if Zero,
cbnz	Compare and Branch if Nonzero,
tbz	Test Bit and Branch if Zero, and
tbnz	Test Bit and Branch if Nonzero.

```
cb{n}z  Rt, <label>
tb{n}z  Rt, #imm6, <label>
```

Ejercicio

- Considere x0=10 y x1=3, escriba las líneas necesarias para calcular: 10 mod 3 o 10%3

$$x\ mod\ n\ =\ x\ -\ (\left\lfloor\frac{x}{n}\right\rfloor *\ n)$$

```
.global _start

_start:
    mov x0, -1          // x
    mov x1, 3           // n

    sdiv x2, x0, x1     // x/n
    mul x2, x2, x1      // x/n * n
    sub x2, x0, x2      // x - (x/n * n)

    mov x0, x2
    mov x8, 93
    svc 0
```

Bibliografía

Arm Limited. (2024). Arm Architecture Reference Manual: for A-profile architecture.

Patterson. D. & Hennesy, J. (2017). Computer Organization and Design: ARM Edition. Elsevier.

Pyeatt, L. & Ughetta, W. (2020). ARM 64-bit Assembly Language. Elsevier.

Smith, S. (2020). Programming with 64-bit ARM Assembly Language. Apress.