09 Funciones y estructuras

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

Conceptos

- Una función es segmento de código que se desea reutilizar.
- Las funciones tienen entradas llamadas argumentos y salida llamada valor de retorno.
- La función que llama a otra es conocida como caller, y la función que es llamada es conocida como callee.

Conceptos

- Por convención los argumentos y valores de retorno se colocan en los registros x0-x7.
- La callee coloca el valor de retorno en X0.
- La caller guarda la dirección de retorno en LR al mismo tiempo que salta a la callee mediante un Branch con BL.

Conceptos

- La llamada se hace mediante BL y el retorno mediante RET.
- La callee no debe modificar los registros x19-x29, LR y el Stack (a menos que se solicite memoria).
- La callee no debe interferir en la caller.

Ejemplo

- Escribir un programa que calcule e imprima el factorial de un número, por ejemplo de 5 sería 120, y crear dos funciones: una llamada factorial que calcule el valor y lo devuelva; una segunda llamada itoa que tome dicho valor y lo convierta a cadena.

```
.global _start

.extern factorial
.extern itoa

.data
buffer: .space 11

.text
_start:
    mov x0, 5           // number
    bl factorial        // x0 = factorial(number)

    ldr x1, =buffer     // load buffer
    bl itoa             // buffer = itoa(x0)

    // print buffer
    mov x0, 1
    ldr x1, =buffer
    mov x2, 11
    mov x8, 64
    svc 0

    // exit
    mov x8, 93
    svc 0
```

```
.global factorial

factorial:
    // initialization
    mov x9, x0          // number
    mov x10, 1          // neutral value


    // iterative multiplication
loop:
    cbz x9, end_loop    // if num == 0 end
    mul x10, x10, x9    // 2i num *= (num-1)
    add x9, x9, -1      // num--
    b loop

end_loop:
    mov x0, x10         // factorial(number)
    ret
```

```
.global itoa

itoa:
    // initialization
    mov x9, x0          // number
    mov x10, x1         // buffer
    mov x11, 10         // base 10
    cbz x9, zero        // handle zero

    // count digits
    mov x12, x9         // x9 saved
    mov x13, 1          // counter
count:
    cbz x12, end_count  // if x12==0 end
    udiv x12, x12, x11  // num = num / 10
    add x13, x13, 1     // counter++
    b count
end_count:
    add x10, x10, x13   // store at end digits
    strb wzr, [x10]     // '\0'
    add x10, x10, -1
    mov w14, 10         // '\n'
    strb w14, [x10]
    add x10, x10, -1    // ready to store number
```

```
    // put last digit in buffer
loop:
    udiv x12, x9, x11   // x12 = num / 10
    mul x13, x12, x11   // x13 = x12 * 10
    sub x13, x9, x13    // remainder
    add x13, x13, '0'   // ascii remainder
    strb w13, [x10]     // store ascii
    add x10, x10, -1    // offset --
    mov x9, x12         // num = num / 10
    cbnz x9, loop       // while num != 0
    ret

    // special case zero
zero:
    mov x9, '0'         // '0'
    strb w9, [x10, 0]
    mov x9, 10          // '\n'
    strb w9, [x10, 1]
    strb wzr, [x10, 2]  // '\0'
    ret
```