Concurso Arquitectura

Arquitectura, Sistemas Operativos y Redes

Dámian Ariel Marotte

Enunciado

Una forma de imprimir un valor entero es realizando una llamada a la función printf. Esta toma como primer argumento una cadena de C (las cuales se representan como un puntero a caracter) indicando el formato y luego una cantidad variable de argumentos que serán impresos. La signatura en C es la siguiente:

```
int printf(const char *format, ...);
```

Enunciado

Una forma de imprimir un valor entero es realizando una llamada a la función printf. Esta toma como primer argumento una cadena de C (las cuales se representan como un puntero a caracter) indicando el formato y luego una cantidad variable de argumentos que serán impresos. La signatura en C es la siguiente:

```
int printf(const char *format, ...);
```

La forma de llamarla en ensamblador es como sigue:

```
.data
format: .asciz "%ld\n"
i:      .quad 0xDEADBEEF

.text
.global main
main:
    movq $format, %rdi # El primer argumento es el formato.
    movq $1234, %rsi   # El valor a imprimir.
    xorq %rax, %rax    # Cantidad de valores de punto flotante.
    call printf
    ret
```

Agregue más llamadas a `printf` en el código para imprimir:

a) El valor del registro rsp.

b) La dirección de la cadena de formato.

c) La dirección de la cadena de formato en hexadecimal.

d) El quad en el tope de la pila.

e) El quad ubicado en la dirección rsp + 8.

f) El valor i.

g) La dirección de i.

Apartado a

```
a:  # "El valor del registro rsp."
    movq $format, %rdi
    movq %rsp, %rsi
    xorq %rax, %rax
    call printf
```

Apartado a

```
a:  # "El valor del registro rsp."
    movq $format, %rdi
    movq %rsp, %rsi
    xorq %rax, %rax
    call printf
```

Apartado b

```
b:  # "La direccion de la cadena de formato."
    movq $format, %rdi
    movq $format, %rsi
    xorq %rax, %rax
    call printf
```

Apartado c

```
c:  # "La direccion de la cadena de formato en hexadecimal."
    movq $formatx, %rdi
    movq $format, %rsi
    xorq %rax, %rax
    call printf
```

Apartado c

```
c:  # "La direccion de la cadena de formato en hexadecimal."
    movq $formatx, %rdi
    movq $format, %rsi
    xorq %rax, %rax
    call printf
```

Apartado d

```
d:  # "El quad en el tope de la pila."
    movq $format, %rdi
    movq (%rsp), %rsi
    xorq %rax, %rax
    call printf
```

Apartado e

```
e:  # "El quad ubicado en la direccion rsp + 8."
    movq $format, %rdi
    movq 8(%rsp), %rsi
    xorq %rax, %rax
    call printf
```

Apartado e

```
e:  # "El quad ubicado en la direccion rsp + 8."
    movq $format, %rdi
    movq 8(%rsp), %rsi
    xorq %rax, %rax
    call printf
```

Apartado f

```
f:  # "El valor de i."
    movq $format, %rdi
    movq i, %rsi
    xorq %rax, %rax
    call printf
```

Apartado g

```
g:  # "La direccion de i."
    movq $format, %rdi
    movq $i, %rsi
    xorq %rax, %rax
    call printf
```

Codigo final

```
.data
    format:  .asciz "%ld\n"
    formatx: .asciz "%#x\n"
    i:       .quad  0xDEADBEEF

.text
    .global main
    main:
        a:  # "El valor del registro rsp."
            movq $format, %rdi
            movq %rsp, %rsi
            xorq %rax, %rax
            call printf

        b:  # "La direccion de la cadena de formato."
            movq $format, %rdi
            movq $format, %rsi
            xorq %rax, %rax
            call printf
```

Codigo final

```
c:  # "La direccion de la cadena de formato en hexadecimal."
    movq $formatx, %rdi
    movq $format, %rsi
    xorq %rax, %rax
    call printf

d:  # "El quad en el tope de la pila."
    movq $format, %rdi
    movq (%rsp), %rsi
    xorq %rax, %rax
    call printf

e:  # "El quad ubicado en la direccion rsp + 8."
    movq $format, %rdi
    movq 8(%rsp), %rsi
    xorq %rax, %rax
    call printf
```

Codigo final

```
    f:  # "El valor de i."
        movq $format, %rdi
        movq i, %rsi
        xorq %rax, %rax
        call printf

    g:  # "La direccion de i."
        movq $format, %rdi
        movq $i, %rsi
        xorq %rax, %rax
        call printf
ret
```