Plancha 2 Ejercicios 5, 9 y 10 Representación Computacional de Datos
Arquitectura del Computador

Hedman Ulises, Pitinari Tomás y Quintero Iago

fact.s

```
.global fact1
fact1:
    movq %rdi, %rax
    movq %rdi, %rbx
    subq $1, %rbx
    jz fin1
    jmp for
for:
    mulq %rbx
    subq $1, %rbx
    jz fin1
    jmp for
fin1:
    ret

.global fact2
fact2:
    movq $1, %rax
factAux:
    pushq %rdi
    decq %rdi
    cmpq $0, %rdi
    jz fin2
    call factAux
fin2:
    popq %rdi
    mulq %rdi
    ret
```

main.c

```
#include <stdio.h>
unsigned long fact1(unsigned long);
unsigned long fact2(unsigned long);
int main(void){
    unsigned long x;
    scanf("%lu", &x);
    printf("fact1: %lu\n", fact1(x));
    printf("fact2: %lu\n", fact2(x));
    return 0;
}
```

9)

```
.data
fmt: .string "%d"
entero: .long -100
funcs:
```

```
        .quad f1
        .quad f2
        .quad f3
.text

f1: movl $0, %esi; movq $fmt, %rdi; call printf; jmp fin
f2: movl $1, %esi; movq $fmt, %rdi; call printf; jmp fin
f3: movl $2, %esi; movq $fmt, %rdi; call printf; jmp fin

.global main

main:
pushq %rbp
movq %rsp, %rbp
# Leemos el entero.
movq $entero, %rsi
movq $fmt, %rdi
xorq %rax, %rax
call scanf
xorq %rax, %rax

#COMPLETE CON DOS INSTRUCCIONES.
movl (entero), %ebx
movq funcs(,%rbx,8), %rdx

jmp *%rdx
fin:
movq %rbp, %rsp
popq %rbp
ret
```

10)

```
.data

.text
# setJump espera recibir una direccion de memoria apuntando a 2 palabras, en la primera
# almacenamos la direccion de la instruccion que llamo a setJump y en la segunda
# se almacenara el valor de retorno
.global setjmp2
setjmp2:
        popq %rbx
        movq %rbx, %rdi
        movq 8(%rdi), %rax
        jmp *%rbx

# longJump espera recibir la misma direccion de memoria apuntando a 2 palabras, guarda
# en la segunda palabra el valor de retorno y salta a la direccion
# especificada en la 1er palabra
.global longjmp2
longjmp2:
        popq %rbx
        movq %rsi, 8(%rdi)
        jmp *%rdi
```