

**P1 - Informe Previo:**

**Subrutinas**

Arquitectura de Computadors

**Daniel Aagaard Pérez i Pau Martín Nadal**

**Eva Marin Tordera**

**I4511**

**1. Implementa en C una función, potencia(int x, int y) (Asumimos que x puede ser positiva o negativa, pero y siempre es positivo). En esta función x se multiplica por sí mismo y a veces. Si y=0 la función debe devolver 1. No debe ser una función recursiva sino iterativa.**

int potencia(int x, int y){

int r = 1;

if(y != 0){

for(unsigned int i = 0; i < y; i++){

r \*= x;

}

}

return r;

}

**2. Suponiendo que tienes implementada la función anterior potencia(x,y), el siguiente programa principal utilizará la función potencia para calcular ax2+bx+c.**

*main() {*

*register int A, B, C, x;*

*int y;*

*A=10; // estos valores los podréis cambiar*

*B=6; // estos valores los podréis cambiar*

*C=-1; // estos valores los podréis cambiar*

*x=3; // estos valores los podréis cambiar*

*y= A \* potencia(x, 2) + B \* x + C; // guardar en memoria el resultado*

*}*

**Traduce a ensamblador el programa main().**

.data

y: .word 0

.align 2

.text

.globl main

main:

addi $sp, $sp, -4

sw $ra, 0($sp)

addi $a0, $zero, 1 # a0 = x = 1

addi $a1, $zero, 2 # a1 = 2, per fer la potencia

jal potencia # cridem a la funció amb $a0 i $a1 i ho retorna a $v0

addi $t0, $zero, -3 # t0 = A = -3

addi $t1, $zero, 2 # t1 = B = 2

addi $t2, $zero, 1 # t2 = C = 1

mul $t3, $t1, $a0 # t3 = B\*x

mul $t0, $t0, $v0 # t0 = A\*potencia(x,2)

add $t0, $t0, $t3 # A\*potencia(x,2) + B\*x

add $t0, $t0, $t2 # A\*potencia(x,2) + B\*x + C

sw $t0, y # Guardem el resultat en la y.

lw $ra, 0($sp)

addi $sp, $sp, 4

jr $ra

.end main

**3. Supongamos el siguiente código en C de una función recursiva que también calcula la potencia (Asumimos que x puede ser positiva o negativa, pero y siempre es positivo):**

*int potencia\_recursiva(int x, int y) {*

*int result;*

*if(y==0) result=1;*

*f(y==1) result=x;*

*else {*

*result = potencia\_recursiva (x ,y/2) \* potencia\_recursiva(x, y-(y/2));*

*}*

*return result;*

*}*

**Donde la llamamos con un main:**

*main(){*

*int result;*

*register int x=2;*

*register int y=5;*

*result=potencia\_recursiva(2,5); // guardar en memoria*

*}*

**Recuerda que se ha de guardar también en la pila la @ de retorno, $ra; así como los argumentos $a0 y $a1 si es necesario cada vez que llamamos a la función.**

**3.1. Tradúcelo a ensamblador:**

.data

result: .word 0

.text

.globl main

main:

addi $sp, $sp, -4

sw $ra, 0($sp)

addi $a0, $zero, 2 # x = 2

addi $a1, $zero, 5 # y = 5

jal potencia\_recursiva

sw $v0, result # guardem a memoria

lw $ra, 0($sp)

addi $sp, $sp, 4

jr $ra

.end main

potencia\_recursiva:

addi $sp, $sp, -12 # fem espai dos arguments i l’adreça de retorn

sw $ra, 8($sp)

sw $a0, 4($sp)

sw $a1, 0($sp)

bne $a1, $zero, cond2 # if(y != 0) cond 2

addi $v0, $zero, 1 # v0 = 1

addi $sp, $sp, 12 # pop 3 items de la pila

jr $ra

cond2: addi $t0, $zero, 1

bne $t0, $a1, else

add $v0, $zero, $a0

addi $sp, $sp, 12 # pop 3 items de la pila

jr $ra

else:

srl $a1, $a1, 1 # y = y >> 1 // y = y/2

jal potencia\_recursiva

lw $t0, 0($sp) # t0 = y

addi $sp, $sp, -4 # push $v0

sw $v0, 0($sp) # guardem $v0 a la pila

srl $a1, $t0, 1 # a1 = y/2

sub $a1, $t0, $a1 # a1 = y-(y/2)

jal potencia\_recursiva

lw $t1, 0($sp) # t1 = potencia\_recursiva(x,y/2)

addi $sp, $sp, 4

mul $v0, $t1, $v0 #v0=potencia\_recursiva(x,y/2)\*potencia\_recursiva(x,y-(y/2))

lw $ra, 8($sp)

addi $sp, $sp, 12

jr $ra