



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Politècnica Superior d'Enginyeria
de Vilanova i la Geltrú



P1 - Informe Final: Subrutinas

Arquitectura de Computadors

Daniel Aagaard Pérez i Pau Martín Nadal

**Eva Marin Tordera
I4511**

1. Traduce al ensamblador la rutina potencia(x,y) e impleméntala en el simulador junto con el programa main de la pregunta 2 del informe previo también en el ensamblador. Simúlalo paso a paso. NOTA: Aunque no sea una rutina que llame a otra ni tampoco recursiva seguiremos la metodología de al entrar en una subrutina guardar en la pila todos los registros \$s0-s7 que vayamos a utilizar. Además si fuese una subrutina que llama a otra o a sí misma ha de guardar también en la pila la @ de retorno, \$ra; así como los argumentos \$a0 y \$a1 si es necesario, por ejemplo en una recursiva.

1.1. Escribe aquí la rutina potencia(x, y) en ensamblador:

```
.data
    y: .word 0
.align 2
.text
.globl main
main:
    addi $sp, $sp, -4
    sw $ra, 0($sp)

    addi $a0, $zero, 3 # a0 = x = 3
    addi $a1, $zero, 2 # a1 = 2, per fer la potencia
    jal potencia        # cridem a la funció amb $a0 i $a1 i retorna a $v0
    addi $t0, $zero, 10 # t0 = A = 10
    addi $t1, $zero, 6  # t1 = B = 6
    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

potencia:
    addi $sp, $sp, -4
    sw $ra, 0($sp)
    addi $v0, $zero, 1
    beq $a1, $zero, fi
    add $t1, $zero, $zero
```

```

for:    slt $t0, $t1, $a1
        beq $t0, $zero, fi
        mul $v0, $v0, $a0
        addi $t1, $t1, 1
        bne $t0, $zero, for
fi:     lw $ra, 0($sp)
        addi $sp, $sp, 4
        jr $ra

```

```

PC      = 400020
EPC     = 0
Cause   = 0
BadVAddr = 0
Status  = 3000fff0

HI      = ffffffff
LO      = ffffffff

R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = a
R3 [v1] = 0
R4 [a0] = 1
R5 [a1] = 2
R6 [a2] = 7ffff9c0
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 2
R10 [t2] = 1
R11 [t3] = 2
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [x0] = 0
R27 [x1] = 0
R28 [gp] = 0
R29 [sp] = 7ffff9b4
R30 [s8] = 0
R31 [ra] = 400018

User Text Segment [00400000]..[00440000]
[00400000] 8fa00000 lw $4, 0($29) ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
[00400014] 0c100009 jal 0x00400024 [main] ; 188: jal main
[00400018] 00000000 nop ; 189: nop
[0040001c] 3402000a ori $2, $0, 10 ; 191: li $v0 10
[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)
[00400024] 23bdf0fc addi $29, $29, -4 ; 7: addi $sp, $sp, -4
[00400028] afbf0000 sw $31, 0($29) ; 8: sw $ra, 0($sp)
[0040002c] 20040001 addi $4, $0, 1 ; 10: addi $a0, $zero, 1 # a0 = x = 1
[00400030] 20050002 addi $5, $0, 2 ; 11: addi $a1, $zero, 2 # a1 = 2, per fer la potencia
[00400034] 0c10001a jal 0x00400068 [potencia] ; 12: jal potencia # cridem a la funció amb $a0 i $a1 i ho retorna a $v0
[00400038] 2008fffd addi $8, $0, -3 ; 13: addi $t0, $zero, -3 # t0 = A = -3
[0040003c] 20090002 addi $9, $0, 2 ; 14: addi $t1, $zero, 2 # t1 = B = 2
[00400040] 200a0001 addi $10, $0, 1 ; 15: addi $t2, $zero, 1 # t2 = C = 1
[00400044] 71245802 mul $11, $9, $4 ; 17: mul $t3, $t1, $a0 # t3 = B*x
[00400048] 71024002 mul $8, $8, $2 ; 18: mul $t0, $t0, $v0 # t0 = A*potencia(x,2)
[0040004c] 010b4020 add $8, $8, $11 ; 19: add $t0, $t0, $t3 # A*potencia(x,2) + B*x
[00400050] 010a4020 add $8, $8, $10 ; 20: add $t0, $t0, $t2 # A*potencia(x,2) + B*x + C
[00400054] 3c011001 lui $1, 4097 ; 21: sw $t0, y # Guardem el resultat en la y.
[00400058] ac280000 sw $8, 0($1)
[0040005c] 8fbf0000 lw $31, 0($29) ; 23: lw $ra, 0($sp)
[00400060] 23bd0004 jr $31 ; 24: addi $sp, $sp, 4
[00400064] 03e00008 jr $31 ; 25: jr $ra
[00400068] 23bdf0fc addi $29, $29, -4 ; 30: addi $sp, $sp, -4
[0040006c] afbf0000 sw $31, 0($29) ; 31: sw $ra, 0($sp)
[00400070] 20020001 addi $2, $0, 1 ; 32: addi $v0, $zero, 1
[00400074] 10a00007 beq $5, $0, 28 [fi-0x00400074]; 33: beq $a1, $zero, fi
[00400078] 00004820 add $9, $0, $0 ; 34: addi $t1, $zero, $zero
[0040007c] 0125402a slt $8, $9, $5 ; 35: slt $t0, $t1, $a1
[00400080] 11000004 beq $8, $0, 16 [fi-0x00400080]; 36: beq $t0, $zero, fi
[00400084] 70441002 mul $2, $2, $4 ; 37: mul $v0, $v0, $a0
[00400088] 21290001 addi $9, $9, 1 ; 38: addi $t1, $t1, 1
[0040008c] 1500fffc bne $8, $0, -16 [for-0x0040008c]
[00400090] 8fbf0000 lw $31, 0($29) ; 40: lw $ra, 0($sp)
[00400094] 23bd0004 addi $29, $29, 4 ; 41: addi $sp, $sp, 4
[00400098] 03e00008 jr $31 ; 42: jr $ra

```

1.2. Prueba con diferentes valores de A,B,C y X: A=10, B=6, C=-1, X=3, A=-3, B=2, C=1, X=1, A=10, B=-3, C=2 y X=0; cuando funcione avisa al profesor.

A=10, B=6, C=-1, X=3 → 107 → 0x6B

```

User data segment [10000000]..[10040000]
[10000000]..[1000ffff] 00000000
[10010000] 0000006b 00000000 00000000 00000000
[10010010]..[1003ffff] 00000000

```

A=-3, B=2, C=1, X=1 → 0 → 0x00

```

User data segment [10000000]..[10040000]
[10000000]..[1003ffff] 00000000

```

A=10, B=-3, C=2, X=0 → 2 → 0x02

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff] 00000000
[10010000] 00000002 00000000 00000000 00000000 . . . . .
[10010010]..[1003ffff] 00000000
```

2. A partir del ejercicio 3 del trabajo previo, implementa el main y la subrutina en el simulador.

a) En el archivo ejercicio3.s debes completar la traducción. Simúlalo paso a paso.

```
.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

```

```

PC      = 400034
EPC     = 0
Cause   = 0
BadAddr = 0
Status  = 3000ff10

HI      = 0
LO      = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 2
R5 [a1] = 5
R6 [a2] = 7ffff9c0
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [t8] = 0
R17 [t9] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [a0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff9b0
R30 [s8] = 0
R31 [ra] = 400018

User Text Segment [00400000]..[00440000]
[00400000] 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
[00400014] 0c100009 jal 0x00400024 [main] ; 188: jal main
[00400018] 00000000 nop ; 189: nop
[0040001c] 3402000a ori $2, $0, 10 ; 191: li $v0 10
[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)
[00400024] 23bdfffc addi $29, $29, -4 ; 6: addi $sp, $sp, -4
[00400028] afbf0000 sw $31, 0($29) ; 7: sw $ra, 0($sp)
[0040002c] 20040002 addi $4, $0, 2 ; 8: addi $a0, $zero, 2 #x = 2
[00400030] 20050005 addi $5, $0, 5 ; 9: addi $a1, $zero, 5 #y = 5
[00400034] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400038] 3c011001 lui $1, 4097 ; 11: sw $v0, result #guardem a memoria
[0040003c] ac220000 sw $2, 0($1)
[00400040] 8fbd0000 lw $31, 0($29) ; 12: lw $ra, 0($sp)
[00400044] 23bd0004 addi $29, $29, 4 ; 13: addi $sp, $sp, 4
[00400048] 03e00008 jr $31 ; 14: jr $ra
[0040004c] 23bdfffc addi $29, $29, -12 ; 19: addi $sp, $sp, -12 # fem espai pels dos arguments i 1 adrea de retorn
[00400050] afbf0008 sw $31, 8($29) ; 20: sw $ra, 8($sp)
[00400054] afa40004 sw $4, 4($29) ; 21: sw $a0, 4($sp)
[00400058] afa50000 sw $5, 0($29) ; 22: sw $a1, 0($sp)
[0040005c] 14a00004 bne $5, $0, 16 [cond2-0x0040005c]
[00400060] 20020001 addi $2, $0, 1 ; 25: addi $v0, $zero, 1 #v0 = 1
[00400064] 23bd000c addi $29, $29, 12 ; 26: addi $sp, $sp, 12 #pop 3 items from stack
[00400068] 03e00008 jr $31 ; 27: jr $ra
[0040006c] 20080001 addi $8, $0, 1 ; 29: addi $t0, $zero, 1
[00400070] 15050004 bne $8, $5, 16 [else-0x00400070]
[00400074] 00041020 addi $2, $0, $4 ; 31: add $v0, $zero, $a0
[00400078] 23bd000c addi $29, $29, 12 ; 32: addi $sp, $sp, 12 #pop 3 items from stack
[0040007c] 03e00008 jr $31 ; 33: jr $ra
[00400080] 00052842 srl $5, $5, 1 ; 36: srl $a1, $a1, 1 # y = y >> 1 // y = y/2
[00400084] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400088] 8fa80000 lw $8, 0($29) ; 38: lw $t0, 0($sp) #t0 = y
[0040008c] 23bdfffc addi $29, $29, -4 ; 39: addi $sp, $sp, -4 #push $v0
[00400090] afa20000 sw $2, 0($29) ; 40: sw $v0, 0($sp) #save $v0 to the stack
[00400094] 00082842 srl $5, $8, 1 ; 41: srl $a1, $t0, 1 #a1 = y/2
[00400098] 01052822 sub $5, $8, $5 ; 42: sub $a1, $t0, $a1 #a1 = y-(y/2)
[0040009c] 0c100013 jal 0x0040004c [potencia_recursiva]
[004000a0] 8fa90000 lw $9, 0($29) ; 44: lw $t1, 0($sp) #t1 = potencia_recursiva(x,y/2)
[004000a4] 23bd0004 addi $29, $29, 4 ; 45: addi $sp, $sp, 4
[004000a8] 71221002 mul $2, $9, $2 ; 46: mul $v0, $t1, $v0 #v0 = potencia_recursiva(x,y/2) *potencia_recursiva(x,y-(y/2))

```

PC = 400054
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 2
R5 [a1] = 2
R6 [a2] = 7ffff9c0
R7 [a3] = 0
R8 [t0] = 1
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff998
R30 [s8] = 0
R31 [ra] = 400088

PC = 4000b4
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 4

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 2
R5 [a1] = 1
R6 [a2] = 7ffff9c0
R7 [a3] = 0
R8 [t0] = 1
R9 [t1] = 2
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff9a4
R30 [s8] = 0
R31 [ra] = 400088

```

User Text Segment [00400000]..[00440000]
[00400000] 8fa40000 lw $4, 0($29)           ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004 addiu $5, $29, 4      ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004 addiu $6, $5, 4      ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080 sll $2, $4, 2        ; 186: sll $v0 $a0 2
[00400010] 00c23021 addu $6, $6, $2      ; 187: addu $a2 $a2 $v0
[00400014] 0c100009 jal 0x00400024 [main] ; 188: jal main
[00400018] 00000000 nop                ; 189: nop
[0040001c] 3402000a ori $2, $0, 10       ; 191: li $v0 10
[00400020] 0000000c syscall              ; 192: syscall # syscall 10 (exit)
[00400024] 23bdfffc addi $29, $29, -4     ; 6: addi $sp, $sp, -4
[00400028] afbf0000 sw $31, 0($29)        ; 7: sw $ra, 0($sp)
[0040002c] 20040002 addi $4, $0, 2        ; 8: addi $a0, $zero, 2 #x = 2
[00400030] 20050005 addi $5, $0, 5        ; 9: addi $a1, $zero, 5 #y = 5
[00400034] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400038] 3c011001 lui $1, 4097         ; 11: sw $v0, result #guardem a memoria
[0040003c] ac220000 sw $2, 0($1)         ; 12: sw $ra, 0($sp)
[00400040] 8fbf0000 lw $31, 0($29)        ; 13: addi $sp, $sp, 4
[00400044] 23bd0004 addi $29, $29, 4      ; 14: jr $ra
[00400048] 03e00008 jr $31              ; 19: addi $sp, $sp, -12 # fem espai pels dos arguments i 1 adrea de retorn
[0040004c] 23bdfffc addi $29, $29, -12    ; 20: sw $ra, 8($sp)
[00400050] afbf0008 sw $31, 8($29)        ; 21: sw $a0, 4($sp)
[00400054] afa40004 sw $4, 4($29)        ; 22: sw $a1, 0($sp)
[00400058] afa50000 sw $5, 0($29)        ; 25: addi $v0, $zero, 1 #v0 = 1
[0040005c] 14a00004 bne $5, $0, 16 [cond2-0x0040005c]
[00400060] 20020001 addi $2, $0, 1        ; 26: addi $sp, $sp, 12 #pop 3 items from stack
[00400064] 23bd000c addi $29, $29, 12     ; 27: jr $ra
[00400068] 03e00008 jr $31              ; 29: addi $t0, $zero, 1
[0040006c] 20080001 addi $8, $0, 1        ; 31: add $v0, $zero, $a0
[00400070] 15050004 bne $8, $5, 16 [else-0x00400070]
[00400074] 00041020 addi $2, $0, $4        ; 32: addi $sp, $sp, 12 #pop 3 items form stack
[00400078] 23bd000c addi $29, $29, 12     ; 33: jr $ra
[0040007c] 03e00008 jr $31              ; 36: srl $a1, $a1, 1 # y = y >> 1 // y = y/2
[00400080] 00052842 srl $5, $5, 1        ; 38: lw $t0, 0($sp) #t0 = y
[00400084] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400088] 8fa80000 lw $8, 0($29)        ; 39: addi $sp, $sp, -4 #push $v0
[0040008c] 23bdfffc addi $29, $29, -4     ; 40: sw $v0, 0($sp) #save $v0 to the stack
[00400090] afa20000 sw $2, 0($29)        ; 41: srl $a1, $t0, 1 #a1 = y/2
[00400094] 00082842 srl $5, $8, 1        ; 42: sub $a1, $t0, $a1 #a1 = y-(y/2)
[00400098] 01052822 sub $5, $8, $5        ; 44: lw $t1, 0($sp) #t1 = potencia_recursiva(x,y/2)
[0040009c] 0c100013 jal 0x0040004c [potencia_recursiva]
[004000a0] 8fa90000 lw $9, 0($29)        ; 45: addi $sp, $sp, 4
[004000a4] 23bd0004 addi $29, $29, 4      ; 46: mul $v0, $t1, $v0 #v0 = potencia_recursiva(x,y/2) *potencia_recursiva(x,y-(y/2))
[004000a8] 71221002 mul $2, $9, $2
[004000ac] 8fbf0000 lw $31, 8($29)
[004000b0] 23bd000c addi $29, $29, 12
[004000b4] 03e00008 jr $31
[004000b8] 00000000 nop
[004000bc] 3402000a ori $2, $0, 10
[004000c0] 0000000c syscall
[004000c4] 23bdfffc addi $29, $29, -4
[004000c8] afbf0000 sw $31, 0($29)
[004000cc] 20040002 addi $4, $0, 2
[004000d0] 20050005 addi $5, $0, 5
[004000d4] 0c100013 jal 0x0040004c [potencia_recursiva]
[004000d8] 3c011001 lui $1, 4097
[004000dc] ac220000 sw $2, 0($1)
[004000e0] 8fbf0000 lw $31, 0($29)
[004000e4] 23bd0004 addi $29, $29, 4
[004000e8] 03e00008 jr $31
[004000ec] 23bdfffc addi $29, $29, -12
[004000f0] afbf0008 sw $31, 8($29)
[004000f4] afa40004 sw $4, 4($29)
[004000f8] afa50000 sw $5, 0($29)
[004000fc] 14a00004 bne $5, $0, 16 [cond2-0x004000fc]
[00400100] 20020001 addi $2, $0, 1
[00400104] 23bd000c addi $29, $29, 12
[00400108] 03e00008 jr $31
[0040010c] 20080001 addi $8, $0, 1
[00400110] 15050004 bne $8, $5, 16 [else-0x00400110]
[00400114] 00041020 addi $2, $0, $4
[00400118] 23bd000c addi $29, $29, 12
[0040011c] 03e00008 jr $31
[00400120] 00052842 srl $5, $5, 1
[00400124] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400128] 8fa80000 lw $8, 0($29)
[0040012c] 23bdfffc addi $29, $29, -4
[00400130] afa20000 sw $2, 0($29)
[00400134] 00082842 srl $5, $8, 1
[00400138] 01052822 sub $5, $8, $5
[0040013c] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400140] 8fa90000 lw $9, 0($29)
[00400144] 23bd0004 addi $29, $29, 4
[00400148] 71221002 mul $2, $9, $2
[0040014c] 8fbf0000 lw $31, 8($29)
[00400150] 23bd000c addi $29, $29, 12
[00400154] 03e00008 jr $31
[00400158] 00000000 nop
[0040015c] 3402000a ori $2, $0, 10
[00400160] 0000000c syscall
[00400164] 23bdfffc addi $29, $29, -4
[00400168] afbf0000 sw $31, 0($29)
[0040016c] 20040002 addi $4, $0, 2
[00400170] 20050005 addi $5, $0, 5
[00400174] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400178] 3c011001 lui $1, 4097
[0040017c] ac220000 sw $2, 0($1)
[00400180] 8fbf0000 lw $31, 0($29)
[00400184] 23bd0004 addi $29, $29, 4
[00400188] 03e00008 jr $31
[0040018c] 23bdfffc addi $29, $29, -12
[00400190] afbf0008 sw $31, 8($29)
[00400194] afa40004 sw $4, 4($29)
[00400198] afa50000 sw $5, 0($29)
[0040019c] 14a00004 bne $5, $0, 16 [cond2-0x0040019c]
[004001a0] 20020001 addi $2, $0, 1
[004001a4] 23bd000c addi $29, $29, 12
[004001a8] 03e00008 jr $31
[004001ac] 20080001 addi $8, $0, 1
[004001b0] 15050004 bne $8, $5, 16 [else-0x004001b0]
[004001b4] 00041020 addi $2, $0, $4
[004001b8] 23bd000c addi $29, $29, 12
[004001bc] 03e00008 jr $31
[004001c0] 00052842 srl $5, $5, 1
[004001c4] 0c100013 jal 0x0040004c [potencia_recursiva]
[004001c8] 8fa80000 lw $8, 0($29)
[004001cc] 23bdfffc addi $29, $29, -4
[004001d0] afa20000 sw $2, 0($29)
[004001d4] 00082842 srl $5, $8, 1
[004001d8] 01052822 sub $5, $8, $5
[004001dc] 0c100013 jal 0x0040004c [potencia_recursiva]
[004001e0] 8fa90000 lw $9, 0($29)
[004001e4] 23bd0004 addi $29, $29, 4
[004001e8] 71221002 mul $2, $9, $2
[004001ec] 8fbf0000 lw $31, 8($29)
[004001f0] 23bd000c addi $29, $29, 12
[004001f4] 03e00008 jr $31
[004001f8] 00000000 nop
[004001fc] 3402000a ori $2, $0, 10
[00400200] 0000000c syscall
[00400204] 23bdfffc addi $29, $29, -4
[00400208] afbf0000 sw $31, 0($29)
[0040020c] 20040002 addi $4, $0, 2
[00400210] 20050005 addi $5, $0, 5
[00400214] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400218] 3c011001 lui $1, 4097
[0040021c] ac220000 sw $2, 0($1)
[00400220] 8fbf0000 lw $31, 0($29)
[00400224] 23bd0004 addi $29, $29, 4
[00400228] 03e00008 jr $31
[0040022c] 23bdfffc addi $29, $29, -12
[00400230] afbf0008 sw $31, 8($29)
[00400234] afa40004 sw $4, 4($29)
[00400238] afa50000 sw $5, 0($29)
[0040023c] 14a00004 bne $5, $0, 16 [cond2-0x0040023c]
[00400240] 20020001 addi $2, $0, 1
[00400244] 23bd000c addi $29, $29, 12
[00400248] 03e00008 jr $31
[0040024c] 20080001 addi $8, $0, 1
[00400250] 15050004 bne $8, $5, 16 [else-0x00400250]
[00400254] 00041020 addi $2, $0, $4
[00400258] 23bd000c addi $29, $29, 12
[0040025c] 03e00008 jr $31
[00400260] 00052842 srl $5, $5, 1
[00400264] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400268] 8fa80000 lw $8, 0($29)
[0040026c] 23bdfffc addi $29, $29, -4
[00400270] afa20000 sw $2, 0($29)
[00400274] 00082842 srl $5, $8, 1
[00400278] 01052822 sub $5, $8, $5
[0040027c] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400280] 8fa90000 lw $9, 0($29)
[00400284] 23bd0004 addi $29, $29, 4
[00400288] 71221002 mul $2, $9, $2
[0040028c] 8fbf0000 lw $31, 8($29)
[00400290] 23bd000c addi $29, $29, 12
[00400294] 03e00008 jr $31
[00400298] 00000000 nop
[0040029c] 3402000a ori $2, $0, 10
[00400300] 0000000c syscall
[00400304] 23bdfffc addi $29, $29, -4
[00400308] afbf0000 sw $31, 0($29)
[0040030c] 20040002 addi $4, $0, 2
[00400310] 20050005 addi $5, $0, 5
[00400314] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400318] 3c011001 lui $1, 4097
[0040031c] ac220000 sw $2, 0($1)
[00400320] 8fbf0000 lw $31, 0($29)
[00400324] 23bd0004 addi $29, $29, 4
[00400328] 03e00008 jr $31
[0040032c] 23bdfffc addi $29, $29, -12
[00400330] afbf0008 sw $31, 8($29)
[00400334] afa40004 sw $4, 4($29)
[00400338] afa50000 sw $5, 0($29)
[0040033c] 14a00004 bne $5, $0, 16 [cond2-0x0040033c]
[00400340] 20020001 addi $2, $0, 1
[00400344] 23bd000c addi $29, $29, 12
[00400348] 03e00008 jr $31
[0040034c] 20080001 addi $8, $0, 1
[00400350] 15050004 bne $8, $5, 16 [else-0x00400350]
[00400354] 00041020 addi $2, $0, $4
[00400358] 23bd000c addi $29, $29, 12
[0040035c] 03e00008 jr $31
[00400360] 00052842 srl $5, $5, 1
[00400364] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400368] 8fa80000 lw $8, 0($29)
[0040036c] 23bdfffc addi $29, $29, -4
[00400370] afa20000 sw $2, 0($29)
[00400374] 00082842 srl $5, $8, 1
[00400378] 01052822 sub $5, $8, $5
[0040037c] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400380] 8fa90000 lw $9, 0($29)
[00400384] 23bd0004 addi $29, $29, 4
[00400388] 71221002 mul $2, $9, $2
[0040038c] 8fbf0000 lw $31, 8($29)
[00400390] 23bd000c addi $29, $29, 12
[00400394] 03e00008 jr $31
[00400398] 00000000 nop
[0040039c] 3402000a ori $2, $0, 10
[00400400] 0000000c syscall
[00400404] 23bdfffc addi $29, $29, -4
[00400408] afbf0000 sw $31, 0($29)
[0040040c] 20040002 addi $4, $0, 2
[00400410] 20050005 addi $5, $0, 5
[00400414] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400418] 3c011001 lui $1, 4097
[0040041c] ac220000 sw $2, 0($1)
[00400420] 8fbf0000 lw $31, 0($29)
[00400424] 23bd0004 addi $29, $29, 4
[00400428] 03e00008 jr $31
[0040042c] 23bdfffc addi $29, $29, -12
[00400430] afbf0008 sw $31, 8($29)
[00400434] afa40004 sw $4, 4($29)
[00400438] afa50000 sw $5, 0($29)
[0040043c] 14a00004 bne $5, $0, 16 [cond2-0x0040043c]
[00400440] 20020001 addi $2, $0, 1
[00400444] 23bd000c addi $29, $29, 12
[00400448] 03e00008 jr $31
[0040044c] 20080001 addi $8, $0, 1
[00400450] 15050004 bne $8, $5, 16 [else-0x00400450]
[00400454] 00041020 addi $2, $0, $4
[00400458] 23bd000c addi $29, $29, 12
[0040045c] 03e00008 jr $31
[00400460] 00052842 srl $5, $5, 1
[00400464] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400468] 8fa80000 lw $8, 0($29)
[0040046c] 23bdfffc addi $29, $29, -4
[00400470] afa20000 sw $2, 0($29)
[00400474] 00082842 srl $5, $8, 1
[00400478] 01052822 sub $5, $8, $5
[0040047c] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400480] 8fa90000 lw $9, 0($29)
[00400484] 23bd0004 addi $29, $29, 4
[00400488] 71221002 mul $2, $9, $2
[0040048c] 8fbf0000 lw $31, 8($29)
[00400490] 23bd000c addi $29, $29, 12
[00400494] 03e00008 jr $31
[00400498] 00000000 nop
[0040049c] 3402000a ori $2, $0, 10
[00400500] 0000000c syscall
[00400504] 23bdfffc addi $29, $29, -4
[00400508] afbf0000 sw $31, 0($29)
[0040050c] 20040002 addi $4, $0, 2
[00400510] 20050005 addi $5, $0, 5
[00400514] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400518] 3c011001 lui $1, 4097
[0040051c] ac220000 sw $2, 0($1)
[00400520] 8fbf0000 lw $31, 0($29)
[00400524] 23bd0004 addi $29, $29, 4
[00400528] 03e00008 jr $31
[0040052c] 23bdfffc addi $29, $29, -12
[00400530] afbf0008 sw $31, 8($29)
[00400534] afa40004 sw $4, 4($29)
[00400538] afa50000 sw $5, 0($29)
[0040053c] 14a00004 bne $5, $0, 16 [cond2-0x0040053c]
[00400540] 20020001 addi $2, $0, 1
[00400544] 23bd000c addi $29, $29, 12
[00400548] 03e00008 jr $31
[0040054c] 20080001 addi $8, $0, 1
[00400550] 15050004 bne $8, $5, 16 [else-0x00400550]
[00400554] 00041020 addi $2, $0, $4
[00400558] 23bd000c addi $29, $29, 12
[0040055c] 03e00008 jr $31
[00400560] 00052842 srl $5, $5, 1
[00400564] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400568] 8fa80000 lw $8, 0($29)
[0040056c] 23bdfffc addi $29, $29, -4
[00400570] afa20000 sw $2, 0($29)
[00400574] 00082842 srl $5, $8, 1
[00400578] 01052822 sub $5, $8, $5
[0040057c] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400580] 8fa90000 lw $9, 0($29)
[00400584] 23bd0004 addi $29, $29, 4
[00400588] 71221002 mul $2, $9, $2
[0040058c] 8fbf0000 lw $31, 8($29)
[00400590] 23bd000c addi $29, $29, 12
[00400594] 03e00008 jr $31
[00400598] 00000000 nop
[0040059c] 3402000a ori $2, $0, 10
[00400600] 0000000c syscall
[00400604] 23bdfffc addi $29, $29, -4
[00400608] afbf0000 sw $31, 0($29)
[0040060c] 20040002 addi $4, $0, 2
[00400610] 20050005 addi $5, $0, 5
[00400614] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400618] 3c011001 lui $1, 4097
[0040061c] ac220000 sw $2, 0($1)
[00400620] 8fbf0000 lw $31, 0($29)
[00400624] 23bd0004 addi $29, $29, 4
[00400628] 03e00008 jr $31
[0040062c] 23bdfffc addi $29, $29, -12
[00400630] afbf0008 sw $31, 8($29)
[00400634] afa40004 sw $4, 4($29)
[00400638] afa50000 sw $5, 0($29)
[0040063c] 14a00004 bne $5, $0, 16 [cond2-0x0040063c]
[00400640] 20020001 addi $2, $0, 1
[00400644] 23bd000c addi $29, $29, 12
[00400648] 03e00008 jr $31
[0040064c] 20080001 addi $8, $0, 1
[00400650] 15050004 bne $8, $5, 16 [else-0x00400650]
[00400654] 00041020 addi $2, $0, $4
[00400658] 23bd000c addi $29, $29, 12
[0040065c] 03e00008 jr $31
[00400660] 00052842 srl $5, $5, 1
[00400664] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400668] 8fa80000 lw $8, 0($29)
[0040066c] 23bdfffc addi $29, $29, -4
[00400670] afa20000 sw $2, 0($29)
[00400674] 00082842 srl $5, $8, 1
[00400678] 01052822 sub $5, $8, $5
[0040067c] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400680] 8fa90000 lw $9, 0($29)
[00400684] 23bd0004 addi $29, $29, 4
[00400688] 71221002 mul $2, $9, $2
[0040068c] 8fbf0000 lw $31, 8($29)
[00400690] 23bd000c addi $29, $29, 12
[00400694] 03e00008 jr $31
[00400698] 00000000 nop
[0040069c] 3402000a ori $2, $0, 10
[00400700] 0000000c syscall
[00400704] 23bdfffc addi $29, $29, -4
[00400708] afbf0000 sw $31, 0($29)
[0040070c] 20040002 addi $4, $0, 2
[00400710] 20050005 addi $5, $0, 5
[00400714] 0c100013 jal 0x0040004c [potencia_recursiva]
[00400718] 3c011001 lui $1, 4097
[0040071c] ac220000 sw $2, 0($1)
[00400720] 8fbf0000 lw $31, 0($29)
[00400724] 23bd0004 addi $29, $29, 4
[00400728] 03e00008 jr $31
[0040072c] 23bdfffc addi $29, $29, -12
[00400730] afbf0008 sw $31, 8($29)
[00400734] afa40004 sw $4, 4($29)
[00400738] afa50000 sw $5, 0($29)
[0040073c] 14a00004 bne $5, $0, 16 [cond2-0x0040073c]
[00400740] 20020001 addi $2, $0, 1
[00400744] 23bd000c addi $29, $29, 12
[00400748] 03e00008 jr $31
[0040074c] 20080001 addi $8, $0, 1
[00400750] 15050004 bne $8, $5, 16 [else-0x00400750]
[00400754] 00041020 addi $2, $0, $4
[00400758] 23bd000c addi $29, $29, 12
[0040075c] 03e00008 jr $31
[00400760] 00052842 srl $5, $5, 1
[00400764] 0c1000
```

b) Si el programa main llama a la subrutina potencia_rekursiva con unos valores iniciales de $x=2$ e $y=5$, como está en el enunciado en C, describe la ejecución dinámica del programa, ayúdate simulando paso a paso en el simulador. Avisa al profesor cuando funcione.

Primer ens guardem la direcció de retorn del main, perquè la necessitem després de cridar a la funció de potencia_rekursiva. Després fem els valors de x i y a `$a0` i `$a1` respectivament perquè els necessitem per a la funció potencia_rekursiva.

Un cop dins de potencia_rekursiva, primer fem espai per guardar-nos els dos arguments i la direcció de retorn de la funció, perquè els necessitem i en ser una funció recursiva en tornar-la a cridar podem perdre aquests valors.

En aquest cas com la $y = 5$, llavors saltarem directament al else que divideix la y entre 2 i tornem a cridar a la funció. Això ho fa dos cops i llavors accedeix quan la $y == 1$. Sumem el valor de la x a `$v0` que serà on retornarem el resultat final de la funció.

Llavors tornem a agafar de la pila el valor inicial de y i afegim el resultat de la primera crida recursiva a la pila (`potencia_rekursiva(x,y/2)`), perquè ho necessitem després per calcular el resultat final.

Més tard, preparem les noves variables per a la crida de la funció, com que la $y == 1$ fem la suma del valor de la x a `$v0` que serà on retornarem el resultat final de la funció.

Finalment, recuperem el resultat final de la primera crida recursiva de la pila i el multipliquem pel resultat aconseguit en la segona crida recursiva. Després d'això es faran totes les crides recursives que falten de fer-se.

Al final tornem a agafar la direcció de retorn que es troba a la pila i tornem al main. Guardem el resultat a la variable global resultat i tornem a agafar l'adreça de memòria del main de la pila.

SPIM Version 9.1.24 of August 1, 2023 (final)
Copyright 1990-2023 by James Larus.