

PROBLEMES DE MIPS 32

```
sumSquare:
    subi $sp, $sp, 12
    sw $ra, 8($sp)
    sw $a0, 4($sp)
    addi $a1, $a0, $zero
    jal mult
    lw $ra, 8($sp)
    lw $a0, 4($sp)
    add $v0, $v0, $a1
    addi $sp, $sp, 12
    jr $ra
    # space on stack
    # save ret addr
    # save x
    # save y
    # mult(x,x)
    # call mult
    # restore ret addr
    # restore x
    # mult()*y
    # free stack space
```



PROBLEMES DE MIPS 32

Tal i com s'ha vist a classe de teoria el processador MIPS pot treballar amb **Little** o **Big Endian**. Per resoldre els exercicis, cal suposar que la màquina on es treballa és un MIPS treballant en *Little Endian*.

1.- Què es podrà veure a memòria quan es tecleja el següent bloc de dades?

```
.byte    3, 7, 54, -1, -25
.word    0, -2, -3, 25, 1000
.float   25.38, 78.97, 12.34, -3.141592, -2564.1
.half    15, 45, -30
.double  26.92
.space   7
.byte    47
.double  46.123, -65.4236172
```

2.- Com quedaran disposades a memòria les següents dades:

```
.byte    28
.word    -65
.byte    5, 10
.half    35
.word    19
.half    35
.half    35
.double  46.123
```

Com quedaran les mateixes dades forçant les alineacions a la posició desitjada?

```
.byte    28
.align   3
.word    -65
.byte    5, 10
.align   2
.half    35
.align   5
.half    35
.word    19
.double  46.123
```

Cal observar l'**extensió del signe** al tipus de dada indicat.

3.- En llegir la memòria de dades s'hi troba el següent contingut:

```
@ 0x1000000:    0x63617270
@ 0x1000004:    0x61636974
```

Què hi ha codificat a memòria si se sap que són:

- 8 Bytes** que corresponen a lletres en **codi ASCII de 8 bits**.
- 8 Bytes** que representen enters.
- 2 Words** que representen enters.
- 4 Halfs** que representen enters.
- 2 Floats**.

Si hi hagués codificat un número com a **Double**, seria un número positiu o negatiu?

4.- Què quedarà al registre **\$t3** (que correspon al registre número **\$11**) quan s'executen les següents instruccions:

```
li    $8, -5734
li    $9, 1024
add   $10, $8, $9
abs   $11, $10
```

5.- Què quedarà al registre **\$t4** (que es correspon al registre **\$12**) quan s'executen les següents instruccions:

```
li.s  $f2, -1635.487
li.s  $f3, 1024.23
add.s $f0, $f2, $f3
abs.s $f1, $f0
mfc1.s $12, $f1
```

6.- Codificar en **Hexadecimal** les següents instruccions:

```
a) lw    $8, 0($1)
b) addi  $9, $9, 4
c) add   $8, $18, $8
d) sw    $8, 12($1)
```

7.- Codifiqueu en **Hexadecimal** les següents instruccions suposant que s'han carregat a la posició de memòria **0x00400020**:

```
        .text
        .globl main

main:   sw    $s0, -25($s1)
        lb    $8, 28($9)
        add   $t0, $t1, $t2
        addi  $v0, $v1, 45
eti1:   j     main
        bne  $a2, $a3, eti1
        jr   $31
```

8.- Quines instruccions **MIPS** es corresponen al següent **codi màquina** en **Hexadecimal**:

```
a) 0x3402000a
b) 0x27a50004
c) 0x00041080
d) 0x00c23021
```

9.- En quines instruccions transforma l'assemblador de MIPS les pseudoinstruccions que s'indiquen a continuació:

- a) `li $7, 25`
- b) `move $t6, $t7`
- c) `bge $5, $6, etiqueta`

10.- Què hi haurà a cada un dels registres \$0, \$8 i \$10 després d'executar el següent programa:

```
        .text
        .globl main
main:
        addi $8, $0, 10
        addi $10, $0, 0
bucle:  add   $10, $10, $8
        addi $8, $8, -1
        bne  $8, $0, bucle
fi:
        .end main
```

11.- Després d'executar el següent fragment de codi MIPS:

```
        .data
g:      .word a
a:      .word taula
taula:  .word a, 1, g, taula, 2, 3, 40

        .globl main
        .text
main:
        la   $t3, g
        add  $t1, $0, $t3
        lw   $t2, 4($t1)
        move $t1, $t2
        lw   $t2, 8($t1)
        sw   $0, ($t2)
        .end
```

- a) Es posa a 0 el contingut de l'adreça g.
- b) Es posa a zero el segon element de la taula.
- c) L'adreça de taula valdrà 0.
- d) a = 0.
- e) taula = 0.

12.- Quantes vegades s'executarà la instrucció de l'adreça **etiquetada** amb 'i':

```
        .text
        .globl main
main:
        addi $sp, -12
        sw   $ra, 8($sp)
        li   $t0, 2
        li   $t1, 1
        sw   $t0, 4($sp)
        sw   $t1, 0($sp)
        jal  salvar
        jal  cridal
i:      jal  restaurar
        j    fi
salvar:
        addi $sp, -12
        sw   $ra, 8($sp)
        sw   $t1, 4($sp)
        sw   $t0, 0($sp)
        jr   $ra
        .end salvar
restaurar:
        lw   $ra, 8($sp)
        lw   $t1, 4($sp)
        lw   $t0, 0($sp)
        addi $sp, 12
        jr   $ra
        .end restaurar
cridal:
        addi $t1, $t0, $t1
        jr   $ra
fi:     .end
```

13.- Després d'executar el següent codi, què quedarà a la posició de memòria **dades**?

```
        .data
dades:  .space 4
        .text
        .globl main
main:
        li   $t0, 0x41
        li   $t1, 0
        li   $t3, 4
eti:    beq  $t3, $0, fi
        jal  A
        jal  B
        addi $t3, $t3, -1
        j    eti
        .end main

A:      .ent  A
        sb   $t0, dades($t1)
        addi $t1, $t1, 1
        jr   $ra
        .end A

B:      .ent  B
        addi $t0, $t0, 1
        jr   $ra
        .end B

fi:     .end
```

14.- Què quedarà al registre **\$t1** en finalitzar l'execució del següent programa?

```
        .text
        .globl main

main:   li    $t2, 12
        xori  $t1, $0, 1

inici:  add   $t1, $t1, $t1
        jal  rutina1
        jal  rutina2
        bgtz $t2, inici
        j    fi
        .end main

        .ent  rutina1
rutina1: add   $t1, $t1, $t1
        move $t3, $ra
        jal  rutina2
        move $ra, $t3
        jr   $ra
        .end  rutina1

        .ent  rutina2
rutina2: add   $t1, $t1, $t1
        addi $t2, -5
        jr   $ra
        .end  rutina2

fi:     .end
```

15.- Escriviu un programa que inicialitzi una **matriu unitària** amb **10x10** valors codificats en **coma flotant** a l'adreça de memòria **mat**:

16.- Què hi quedarà als registre **\$8**, **\$9** i **\$10** després d'executar el següent codi dins d'un programa?

```
        addi  $sp, $sp, -12
        sw   $8, 0($sp)
        sw   $9, 4($sp)
        sw   $10, 8($sp)

        lw   $10, 0($sp)
        lw   $8, 4($sp)
        lw   $9, 8($sp)
        addi $sp, $sp, 12
```

17.- Què s'hauria de codificar per a accedir a la posició **mat[i][j]** d'una **matriu de words de 50x90** guardada disposada per columnes consecutives a la posició de memòria **mat**: suposant que **i** és l'índex de **0 a n - 1** guardat a **\$t1** i que **j** és l'índex de **0 a m - 1** guardat a **\$t2**.

18.- Què quedarà al registre **\$t1** després d'executar les instruccions:

```
B:    .word 2
      addi $t2, $0, 10
      add  $t2, $t2, $t2
      lw   $t1, B($0)
      sll  $t1, $t2, 1
      sw   $t2, B($0)
```

- a) 10
- b) 20
- c) 2
- d) 40
- e) 22

19.- Després d'executar el següent codi font en assemblador de **MIPS**:

```
li    $9, 10
xori  $10, $9, 4
subu  $29, $29, 4
sw    $9, 0($29)
sw    $10, a($9)
addi  $11, $10, -4
lw    $10, a($11)
lw    $10, a($29)
addi  $29, $29, 4
```

A l'adreça de memòria **a + 10** hi quedarà:

- a) 10
- b) 14
- c) 20
- d) a
- e) a + 4

20.- Després d'executar el codi de l'exercici anterior, quedarà un 14

- a) Al registre **\$11**.
- b) Al registre **\$9**.
- c) A l'adreça **a + 10**.
- d) A **cap lloc dels utilitzats**.
- e) A la **pila**.

21.- Després de l'execució del codi de l'exercici 19:

- a) El **punter de la pila** seguirà apuntant al mateix lloc.
- b) La **pila** ha quedat **modificada**.
- c) Aquest codi **no** fa servir la **pila**.
- d) El registre **\$29** queda amb un 0.
- e) El registre **\$sp** queda a **\$sp + 4**.

22.- De quina mida és la matriu de **floats** guardada a l'estructura de dades següent:

```
mat: .word    f0, f1, f2, f3, f4
f0:  .space   40
f1:  .space   40
f2:  .space   40
f3:  .space   40
f4:  .space   40
```

- a) 20x4
- b) 5x40
- c) 40x20
- d) 4x5
- e) 5x10

23.- En una matriu de **4x4 enters** guardada **per files consecutives** en una màquina **MIPS** a la posició **mat:**, l'element [2, 3], dada continguda al registre **\$8**, hauria de ser escrita a memòria com:

- a) `lw $8, 24($0)`
- b) `sb mat, $8($0)`
- c) `lb $6, mat($6)`
- d) `sw $8, mat($5)`
- e) `move $8, mat[6]`

24.- Les següents declaracions en llenguatge **assamblador del 386**

```
N    equ    10
M    db     25
P    dw     12
```

Reescriu en llenguatge assamblador de **MIPS** serien:

- | a) | b) | c) | d) | e) |
|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| <code>N = 10</code> | <code>N: .byte 10</code> | <code>N: .byte 10</code> | <code>lw N, 10</code> | <code>N: .word 10</code> |
| <code>M: .byte 25</code> | <code>M: .byte 25</code> | <code>M = 10</code> | <code>lb \$10, 25</code> | <code>M: .word 25</code> |
| <code>P: .word 12</code> | <code>P: .word 12</code> | <code>P = 12</code> | <code>lw P, 12</code> | <code>P: .word 12</code> |

25.- Quina de les següents afirmacions és certa?

- a) Les **adreces absolutes NO queden fixades** fins el moment de **linkar (muntar) el codi**.
- b) Els **fixers objecte** tenen el **codi font en assamblador**.
- c) Les **llibries de fixers objecte** ja no tenen referències a les etiquetes del assamblador.
- d) Els **programes executables**, per a funcionar correctament, no necessiten que es carregui res més a memòria.
- e) Els **fixers objecte** de les llibries estàtiques ja estan instalats a memòria abans de carregar el codi executable.