Tema 6: LLENGUATGE D'ASSEMBLADOR

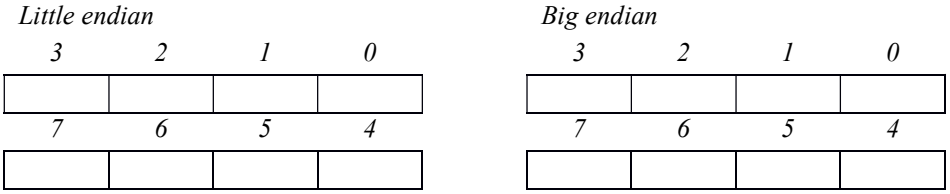
Grau en Informàtica

EXERCICIS

Organització de la memòria

1. Distribuïu les següents dades, de 4 bytes de grandària cada una, a les adreces de memòria corresponents.

Dada 1: 0xABCDEFFF, Dada 2: 0x01234567



2. Donades les següents directives de dades, indiqueu quin serà el contingut de la memòria de dades, sabent que NULL representa el caràcter nul i que la màquina en qüestió emmagatzema les paraules segons el format Little Endian. Indiqueu clarament les zones de memòria de contingut desconegut.

```
.data 0x10000028
.byte 3
.ascii "ABC"
.float 1.5
.word 0xFFFF
.half 19,38
```

31 24	23 16	15 8	7 0	**Adreça**

3. Donades les següents directives de dades, indiqueu quin serà el contingut de la memòria de dades, sabent que NULL representa el caràcter nul i que la màquina en qüestió emmagatzema les paraules segons el format Little Endian. Indiqueu clarament les zones de memòria de contingut desconegut.

```
.data 0x1000000C
.space 2
.half 0xF0
.asciiz "050"
.double 1.5
```

31 24	23 16	15 8	7 0	Adreça

4. Donades les següents directives de dades, indiqueu quin serà el contingut de la memòria de dades, sabent que NULL representa el caràcter nul i que la màquina en qüestió emmagatzema les paraules segons el format Little Endian. Indiqueu clarament les zones de memòria de contingut desconegut.

```
.data 0x10000000
.half 5,3
.byte 3
.data 0x10000011
.byte 5
```

31 24	23 16	15 8	7 0	Adreça

Joc d'instruccions

5. Donat el contingut següent de la memòria de dades:

Memòria de dades	Adreça
0x 6C FF FF FF	0x10000000
0x AB 77 80 44	0x10000004
31 0	

Quin valor tindran els registres $5 i $6 després d'executar-ne les instruccions següents?

```
lui $2, 0x1000
lh $5, 0($2)
lw $6, 4($2)
```

6. Donat el contingut següent de la memòria de dades:

Memòria de dades	Adreça
0x 12 34 56 78	0x10010008
0x CB 00 88 00	0x1001000C
31 0	

Quin valor tindran els registres $4 i $5 després d'executar-ne les següents instruccions?

```
lui $3, 0x1001
lw $4, 12($3)
lb $5, 9($3)
```

Programació en assemblador i codi màquina

7. Donat el codi en llenguatge d'assemblador MIPS R2000 que es mostra a continuació.

```
      .data 0x100000A0
      .byte 1,2,3
      .half 4
dada1:.word 8
dada2:.word 2
      .space 5
      .word 9
      .text 0x00400000
      .globl __start
```

```
__start:
      la $2, dada1
      la $3, dada2
      lw $8,0 ($2)
      lw $4,-4 ($3)
      add $9,$4,$8
      sw $9, 0 ($2)
      .end
```

A. Quin és el contingut de la memòria de dades abans de l'execució del programa?

31 24	23 16	15 8	7 0	Adreça

B. Quin és el contingut de la memòria de dades després de l'execució del programa?

31 24	23 16	15 8	7 0	**Adreça**

C. Quin serà el valor emmagatzemat en els registres següents després de l'execució del programa?

Registre	**Valor**
$2	
$3	
$8	
$4	
$9	

D. Indiqueu la seqüència d'instruccions per les quals es traduiria la pseudoinstrucció `la $2, dada1`

E. Codifiqueu la instrucció `lw $4,-4 ($3)`

8. En un algorisme d'encriptació per blocs, es van xifrant blocs d'una grandària concreta. Per a augmentar la seguretat de l'algoritme, se solen fer operacions prèvies entre blocs. Un dels modes d'operar es coneix com CBC i consisteix a realitzar una OR exclusiva entre el bloc de text a xifrar i el bloc precedent. El codi següent implementa el funcionament descrit.

```
.data 0x10000000
gran:       .half 8                 # Grandària del bloc
BlocA:      .asciiz "or bloqu"      # Bloc precedent
BlocB:      .asciiz "es, se v"      # Bloc a xifrar
BlocF:      .space 8                # Espai per al resultat

.globl __start
.text 0x00400000
__start:
      # Lectura de les dades inicials
      la $10, BlocA     # Llegim l'adreça del bloc precedent
      la $11, BlocB     # Llegim l'adreça del bloc a xifrar
      la $12, BlocF     # Llegim l'adreça del bloc resultat
      la $13, gran            # Llegim la grandària del bloc
      lh $14, 0($13)

      # bucle de l'algorisme
bucle:
      beq $14, $0, fi
      lb $20,0($10)
      lb $21,0($11)
      xor $22,$20,$21
      sb $22,0($12)
      addi $10,$10,1
      addi $11,$11,1
      addi $12,$12,1
      addi $14,$14,-1
      j bucle
fi:   # Final de l'algorisme
.end
```

A. Indiqueu quin contingut tindrà la memòria de dades abans d'executar el programa.

31 24	23 16	15 8	7 0	**Adreça**

B. Indiqueu quin contingut tindrà la memòria de dades després d'executar el programa.

31 24	23 16	15 8	7 0	**Adreça**

C. Quin serà el valor emmagatzemat en els següents registres després de l'execució del programa?

Registre	**Valor**
$10	
$11	
$12	
$13	
$14	

D. Codifiqueu la instrucció `j bucle`

E. Codifiqueu la instrucció `beq $14, $0, fi`

9. Donat el programa següent en assemblador MIPS R2000:

```
.data 0x10000000
estat:     .byte 25
zonaA:     .word xffffffff,0xffffffff, 0xffffffff, 0xffffffff
grandaria: .word 4

.text 0x400400
.globl __start
__start:
        la $6, estat
        lb $6, 0($6)
        la $7, zonaA
        la $8, grandaria
        lw $9, 0($8)
        li $10, 0x00000000
        li $11, 0xaaaaaaaa
        beq $6,$0,accio0
accio1:
        beq $9,$0,fi
        sw $11, 0($7)
        addi $7,$7,4
        addi $9,$9,-1
        j accio1
accio0:
        beq $9,$0,fi
        sw $10, 0($7)
        addi $7,$7,4
        addi $9,$9,-1
        j accio0
fi:
.end
```

A. Quin és el contingut de la memòria de dades abans de l'execució del programa?

31 24	23 16	15 8	7 0	**Adreça**

B. Quin és el contingut de la memòria de dades després de l'execució del programa?

31 24	23 16	15 8	7 0	**Adreça**

C. Codifiqueu la instrucció `j accio0`

D. Codifiqueu la instrucció `beq $6, $0, accio0`

10. Com a part d'un algoritme de realitat virtual, es vol calcular el volum d'un prisma regular i comprovar si aquest volum supera un cert llindar. En el cas que el volum supere el llindar, el programa escriu un 1 en el byte de memòria etiquetat com "res"; en qualsevol altre cas escriu un 0. Així mateix, el volum calculat s'emmagatzema en l'adreça de memòria etiquetada com "vol". El codi proposat és el següent:

```
.data 0x10000000
llargA:     .half 100   # Aresta A
llargB:     .half 50    # Aresta B
llargC:     .half 25    # Aresta C
llindar:    .word 1500  # Llindar per a la comparació
res:        .byte 0     # Espai per al resultat de la comparació
vol:        .word 0     # Espai per al volum del prisma
```

```
.globl __start
.text 0x00400000
__start:
      # Lectura dels valors de les arestes
      la $20, llargA
      lh $10,0($20)
      la $20, llargB
      lh $11,0($20)
      la $20, llargC
      lh $12,0($20)

      # Càlcul de volum
      mult $10,$11
      mflo $13
      mult $12,$13
      mflo $13

      # Emmagatzemem el volum en la memòria
      la $20, vol
      sw $13, 0($20)

      # Comparació amb el llindar
      la  $20, llindar
      lw  $14, 0($20)
      slt $15, $14, $13
      # Emmagatzemem el resultat
      la $20, res
      sb $15, 0($20)
.end
```

A. Indiqueu quin és l'estat del segment de dades abans que s'execute el programa.

31 24	23 16	15 8	7 0	**Adreça**

B. Feu una proposta de nova declaració de dades que reduïsca els espais de memòria no usats a causa de l'alineament

C. Quin serà el valor emmagatzemat en els següents registres després de l'execució del programa? En tot cas, indiqueu en quina base estan expressats els valors numèrics

Registre	Valor
$10	
$11	
$12	
$13	
$14	
$15	
$20	

D. Codifiqueu la instrucció `slt $15, $14, $13`

11. El codi següent conté error: identifiqueu-los.

```
      .data 0x10000000
vector: .byte 0x333, 0x88, 0x54, 0x77
      .word 0x55
      .half 0x44445555
      .space 18
      .text 0x400000
      .globl __start
__start:
      lui $10, 0x1000
      ori $10, $10, 0x0003
      lw $11, 0($10)
      lw $12, 1($10)
.end
```

12. Quin és el resultat d'executar el codi següent?

```
      .text 0x400800
      .globl __start
__start:
bucle: lui $10, 0xFFFF
       andi $10, $10, 0xFFFF
       beq $10, $zero, bucle
       li $10, 0x12345678
.end
```

13. Escriviu un programa en llenguatge d'assemblador del MIPS R2000 que implemente les operacions següents:

resultat = dadaa – dadab + dadac – dadad

Heu de tenir en compte el següent:

- Les dades "dadaa", "dadab, "dadac" i "dadad" es defineixen com enters de 32 bits ubicats a partir de l¡adreça de memòria 0x10002000.
- Heu de reservar espai per a emmagatzemar el resultat com enter de 32 bit a partir de l'adreça de memòria 0x10001000.
- El programa emmagatzemarà el resultat de les operacions en "resultat".

14. Codifiqueu en llenguatge assemblador del MIPS R2000 un programa que realitze la suma de dues variables de tipus *short int* (16 bits) anomenades "dadaa" i "dadab", i emmagatzeme el resultat de la suma en "dadac". Heu de fer la reserva de dades necessària per a accedir a les tres variables, i escriure les instruccions que adients per a realitzar les operacions de suma i emmagatzematge. El codi d'alt nivell següent mostra el que es demana:

```
short int dadaa =  9;
short int dadab = 12;
short int dadac;
dadac = dadaa+dadab;
```

15. Escriviu un programa en llenguatge d'assemblador del MIPS R2000 que calcule l'operació dadac = dada * dadab. Heu de tenir en compte les especificacions següents:

- "dadaa" i "dadab" han de definir-se com enters de 16 bits a partir de l'adreça de memòria 0x10000000.
- Heu de reservar espai per a emmagatzemar el resultat en "dadac" com enter de 32 bit a partir de l'adreça de memòria 0x10001000.

16. Escriviu un programa en llenguatge d'assemblador del MIPS R2000 que realitze l'operació de divisió entera dada / dadab. Heu de tenir en compte les especificacions següents:

- En primer lloc cal comprovar que dadab és distint de zero per continuar amb la divisió. En cas contrari, el programa ha de saltar a l'etiqueta "divisioperzero" .
- El quocient de la divisió ha de guardar-se en la variable "quocient".
- El residu de la divisió ha de guardar-se en la variable "residu".
- Heu de declarar totes les variables "dadaa", "dadab", "quocient" i "residu" com enters de 32 bit ubicats a partir de l'adreça de memòria 0x10000000.

17. Codifiqueu en llenguatge d'assemblador del MIPS R2000 un programa que realitze l'operació resultat = dada+dadab, comprovant si es produeix o no desbordament (overflow) i indicant-ho en la variable "hihadesbordament". Heu de tenir en compte les especificacions següents:

- Heu de declarar totes les variables "dadaa", "dadab", "resultat" i "hihadesbordament" con enters de 32 bit a partir de l'adreça de memòria 0x10000000.
- Després de fer l'operació, s'emmagatzemarà un 1 en "hihadesbordament" si el resultat està fora de rang. En cas contrari s'emmagatzemarà un 0. en "hihadesbordament".
- Per detectar el desbordament heu de comprovar si el bit de signe dels operand és el mateix però diferent del signe del resultat, el que indica que hi ha desbordament.

18. Codifiqueu en llenguatge d'assemblador del MIPS R2000 un programa que realitze l'operació resultat = dada – dadab, comprovant si es produeix o no desbordament (overflow) i indicant-ho en la variable "hihadesbordament". IMPORTANT, el desbordament no es detecta de la mateixa manera que en la suma.Heu de tenir en compte les especificacions següents:

- Heu de declarar totes les variables "dadaa", "dadab", "resultat" i "hihadesbordament" con enters de 32 bit a partir de l'adreça de memòria 0x10000000.
- Després de fer l'operació, s'emmagatzemarà un 1 en "hihadesbordament" si el resultat està fora de rang. En cas contrari s'emmagatzemarà un 0. en "hihadesbordament".
- Per detectar el desbordament heu de comprovar si el bit de signe dels operands és igual. En aquest cas no pot haver-hi desbordament. En cas contrari, hi ha desbordament si el bit de signe del minuend és diferent del bit de signe del resultat.

19. Considerant les directives de dades que es mostren a continuació, escriviu un programa en llenguatge assemblador del MIPS R2000 que guarde en "tira_res" la mateixa cadena de caràcters emmagatzemada en la variable "tira" , però convertint cadascuna de les lletres a minúscules. En la taula ASCCI es pot observar que la diferència entre el codi de una lletra majúscula i el codi de la mateixa lletra minúscula és el valor del bit 5, que per les majúscules és 0 i per las minúscules és 1. Per tant, podem convertir majúscules a minúscules i a l'inrevés sumant-li i restant-li 32, o ficant a 1 i 0 el bit 5.

```
.data 0x10000000
tira: .asciiz "ABC"
tira_res: .space 3
```

Exercicis genèrics

20. Tenint en compte el format d'instrucció vist en les transparències, quantes instruccions de tipus R, I i J pot tindre el MIPS?

21. Responeu a les qüestions següents sobre les instruccions de tipus R

A. Si totes les instruccions de tipus R tenen el codi d'operació 0 quantes instruccions de tipus R com a màxim pot tindre el MIPS?

B. En les instruccions *sll* i *srl*, quin és el desplaçament màxim que es pot posar? Seria interessant tindre més desplaçament?

C. Si el banc de registres del MIPS tinguera 64 registres de 32 bits, quins canvis implicaria en el format de les instruccions de tipus R?

D. Si la grandària dels registres del banc de registres del MIPS passara a ser de 64 bits, mantenint-se 32 registres, quins canvis implicaria en el format de les instruccions de tipus R?

22. Responeu a les següents qüestions sobre les instruccions de tipus I

A. Si el banc de registres del MIPS tinguera 64 registres de 32 bits, quins canvis implicaria en el format de les instruccions de tipus I?

B. Quina és la distància màxima a què pot arribar un salt condicional?

C. Per què es codifica en complement a dos la dada immediata en les instruccions de salt condicional?

D. Per què es codifica el salt en paraules en les instruccions de salt condicional? A quina distància màxima s'arribaria si es codificara en bytes i no en paraules?

E. Quin és el desplaçament màxim teòric on es pot arribar amb una instrucció de càrrega o emmagatzemament?

F. Per què les instruccions de càrrega i emmagatzemament tenen el desplaçament codificat en bytes?

23. Responeu a les següents qüestions sobre les instruccions de tipus J

A. El fet de no poder saltar a una adreça que no comence per 0x0…. fa que el MIPS es deixe fora part de la zona de memòria destinada al codi? Per què?

B. Per què la instrucció *jr* no es considera de tipus J?

C. La grandària d'adreça del MIPS és de 32 bits. En què afectaria les instruccions de tipus J un augment de grandària d'adreça del MIPS a 64 bits?