Parcial EC

Cognoms: ..... Nom: .....
DNI: .....

EXAMEN PARCIAL D'EC
2 de novembre de 2023

- L'examen consta de 6 preguntes, que s'han de contestar als mateixos fulls de l'enunciat. No oblidis posar el teu nom i cognoms a tots els fulls.
- La durada de l'examen és de 1:30 hores (90 minuts)
- Les notes i la solució es publicaran al Racó el 13 de novembre. La revisió es farà presencialment el 14 de novembre a les 10h a 11h a la sala C6-E101.

Pregunta 1 *(1,5 punts)*

Donades les següents declaracions de dades globals en assemblador del MIPS, que s'ubiquen a memòria a partir de l'adreça 0x10010000:

```
        .data
a:      .byte       -1, 1, -5, 5
b:      .dword      0x02
c:      .byte       0x0A
d:      .word       a
```

a) Omple la següent taula amb el contingut de la memòria **EN HEXADECIMAL** (sense el prefix 0x). Les posicions de memòria no ocupades per cap dada es deixen **EN BLANC**.

@Memòria	Dada	@Memòria	Dada	@Memòria	Dada
0x10010000		0x1001000A		0x10010014	
0x10010001		0x1001000B		0x10010015	
0x10010002		0x1001000C		0x10010016	
0x10010003		0x1001000D		0x10010017	
0x10010004		0x1001000E		0x10010018	
0x10010005		0x1001000F		0x10010019	
0x10010006		0x10010010		0x1001001A	
0x10010007		0x10010011		0x1001001B	
0x10010008		0x10012012		0x1001001C	
0x10010009		0x10010013		0x1001001D	

b) Quin és el valor final de `$t0` en hexadecimal després d'executar el següent fragment de codi?

```
        la          $t0, a
        lb          $t0, 2($t0)
        li          $t1, 4
        addu        $t0, $t0, $t1
```

`$t0 =` 0x

c) Quin és el valor final de $t0 en hexadecimal després d'executar el següent fragment de codi?

```
        la          $t0, d
        lw          $t0, 0($t0)
        addiu       $t0, $t0, 8
        lb          $t0, 0($t0)
        ori         $t0, $t0, 1
```

`$t0 =` 0x

Pregunta 2 *(1,5 punts)*

Un programa està compost de dos mòduls que es compilen i assemblen separadament per generar sengles fitxers objecte. Per a generar l'executable cal enllaçar-los després amb el muntador. El codi en C de les funcions main() i func() dels dos mòduls és el següent:

```
MÒDUL 1:    int main(){ sum(W,4); }
MÒDUL 2:    void sum(int *p,int n){ if (n>=0){V[i]= *p+X; sum(p+1,n-1);}}
```

Les variables W, V, X són globals de tipus int. W i V estan definides en el MODUL 1. X està definida al MODUL 2. Hem traduït els dos fitxers a MIPS amb el següent resultat (hem afegit a l'esquerra els números de línia per facilitar les respostes posteriors).

MÒDUL 1

```
1           .data
2
3
4     W:    .word -1, 0, -2, 0, -3
5     V:    .word  0, 0, 0, 0, 0
6
7           .text
8           .globl main
9
10    main: addiu $sp, $sp, -4
11          sw    $ra, 0($sp)
12          la    $a0, W
13          move  $a1, 4
14          jal   sum
15          lw    $ra, 0($sp)
16          addiu $sp, $sp, 4
17          jr    $ra
18
19
20
```

MÒDUL 2

```
1           .data
2
3
4     X:    .word 7
5
6           .text
7
8     sum:  blt   $a1, $zero, fi
9           la    $t0, X
10          lw    $t0, 0($t0)
11          lw    $t1, 0($a0)
12          addu  $t0, $t0, $t1
13          la    $t2, VEC
14          sll   $t3, $a1, 2
15          addu  $t2, $t2, $t3
16          sw    $t0, 0($t2)
17          addiu $a0, $a0, 4
18          addiu $a1, $a1, -1
19          jal   sum
20    fi:   jr    $ra
```

a) Quan hem intentat enllaçar els dos fitxers objecte generats per l'assemblador, l'enllaçador ha detectat diversos errors del tipus "referència no-resolta". ¿Com caldrà corregir o completar el codi MIPS perquè no torni a fallar? (fes servir tantes files de la taula com necessitis)

MÒDUL	LÍNIA	LÍNIA DE CODI CORRECTA

b) Contesta les següents preguntes suposant que s'han tornat a assemblar els dos fitxers després de corregir els errors i que totes les instruccions conserven la numeració de línies original:

Pregunta	MÒDUL 1	MÒDUL 2
Quines etiquetes conté la Taula de Símbols Globals en cada fitxer objecte?		
Quines instruccions en cada fitxer objecte (sols el número de línia) requereixen adreces absolutes (de dades o de codi)?		
Quines instruccions en cada fitxer objecte (sols el número de línia) contenen referències no-resoltes (a dades o a codi definits en l'altre mòdul)?		

Cognoms: ..... Nom: .....
DNI: .....

Pregunta 3 *(2 punts)*

Donades les següents declaracions en C:

```
int *p, y;
int my_function(int x, int *q) {
      // sentència 1
      y = *(q + 3);

      // sentència 2
      p[7] = x;

      // sentència 3
      if ((x < 100) || (x%8 != 0))
            return x>>3;
      else
            return -x;
}
```

a) Tradueix la sentència 1: `y = *(q + 3);`

b) Tradueix la sentència 2: `p[7] = x;`

c) Completa els calaixos buits del següent codi MIPS per tal que sigui la traducció correcta de la sentència 3.

```
et1:                    $t3, $a0, 100
et2:   bne              $t3, $zero,
et3:                    $t4, $a0, 7
et4:                    $t4, $zero, et7
et5:                    $v0, $a0, 3
et6:   b                et8
et7:   subu             $v0,                ,
et8:
```

Pregunta 4 *(1,5 punts)*

El processador FIBEC disposa de 4 tipus d'instruccions diferents: A, B, C i D. La següent taula mostra quin és el número d'instruccions executades per a un programa sota consideració i el CPI de cada tipus d'instrucció. La freqüència de rellotge de el processador és de 4GHz y la potencia dissipada és P=350W.

	Nombre d'instr.	**CPI**
A	$5 \cdot 10^9$	1
B	$4 \cdot 10^9$	3
C	$1 \cdot 10^9$	1
D	$2 \cdot 10^9$	3

a) Calcula el CPI mitjà del programa sota consideració

CPI =

b) Indica quin és el temps d'execució (en segons) del programa

T_{exe} =

c) Calcula l'energia consumida en Joules durant l'execució del programa

E =

Pregunta 5 *(2 punts)*

Donades les declaracions de funcions en C:

```
void f(char v[], int *q);
int  g(char a, int b);

void exam(int *p, int k) {
      char V[10];
      int   n;
      f(V, &n);
      *(p+1) = g(V[k], n);
}
```

Cognoms: ..... Nom: .....
DNI: .....

a) Quin és el nombre mínim de registres segurs que cal utilitzar dins la subrutina exam? Quines dades guardareu a cadascun d'ells?

b) Dibuixa el bloc d'activació de la subrutina `exam` indicant clarament, per a cada element, la seva mida i el desplaçament relatiu al `$sp` necessari per accedi-hi.

c) Completa les caixes buides del següent codi MIPS per tal que sigui la traducció de la funció exam (observa que el pròleg i l'epíleg no s'han de programar). No oblidis d'incloure al lloc apropiat les instruccions que col·loquen en registres segurs els elements que ho requereixin:

```
exam:
      # pròleg on es reserva lloc a la pila i se salven registres segurs
         SE SUPOSA JA PROGRAMAT, NO ESCRIURE RES
      # Traduir la sentència:      f(V, &n);

      # Traduir la sentència: *(p+1) = g(V[k], n);

      # epíleg on s'allibera la pila i es restauren registres segurs
        SE SUPOSA JA PROGRAMAT, NO ESCRIURE RES
      jr    $ra
```

Pregunta 6 *(1,5 punts)*

Considera les següents declaracions en C:

```
int Q[64][64];

void func(int mat[][256], int i, int j) {
      int k;
      for (k=0; k<64: k++) {
            mat[i+k][j+k] = Q[k][63-k];
      }
}
```

El següent fragment de codi conté la traducció de la funció `func`, aplicant-li l'optimització d'accés seqüencial i d'eliminació de la variable d'inducció. En concret, s'han usat els registres `$t0` i `$t1` com a punters per a recórrer les matrius **Q** i **mat** respectivament, i s'ha usat el registre `$t2` per a l'adreça final del punter `$t0`. Completa les instruccions que falten als requadres per tal que la traducció sigui correcta:

```
func:
      la          $t0, [          ]   # @ inicial de $t0
      sll         $t4, $a1, [     ]
      sll         $t5, $a2, [     ]
      addu        $t4, $t4, $t5
      addu        $t1, $a0, $t4       # @ inicial de $t1
      [                       ]       # $t2 = @ final de $t0
for:
      [          ] $t0, $t2, fifor
      lw          $t4, 0($t0)         # accés a Q
      sw          $t4, 0($t1)         # accés a mat
      addiu       $t0, $t0, [     ]
      addiu       $t1, $t1, [     ]
      b           for
fifor:
      jr          $ra
```