COGNOMS: **GRUP:**

NOM:

EXAMEN PARCIAL D'EC GRUP 30
30 d'abril de 2020

L'examen té 7 preguntes, que s'han de contestar als fitxers **RespostaX.txt** (on X és el número de pregunta). No oblidis posar el teu nom i cognoms a la capçalera de cada fitxer de respostes, i pujar-lo al Racó abans que acabi el temps. L'examen comença a les 10:00h i acaba a les 12:00h. Pots pujar cada un dels fitxers tants cops com et faci falta, sols comptarà la darrera versió.

Pregunta 1. (1,25 punts)

Donada la següent funció en C:

```
int f(int v[100])
{
int i, result=0;
      for (i=0; i<100; i++)
          result += v[i];
      return result;
}
```

Hem traduït i optimitzat el codi en assemblador MIPS de la següent manera:

```
f:        addiu   $v0, $zero, 0         # result=0
          addiu   $t1, $zero, 0         # i=0
          addiu   $t2, $zero, 100
for:
          sll     $t3, $t1, 2
          addu    $t3, $a0, $t3
          lw      $t4, 0($t3)           # $t4 = v[i]
          addu    $v0, $v0, $t4         # result += v[i]
          addiu   $t1, $t1, 1           # i++
          slt     $t5, $t1, $t2
          bne     $t5, $zero, for
          jr      $ra                   # aquest salt sempre salta!
```

La taula següent mostra els CPI de cada tipus d'instrucció en un computador MIPS, quan executa l'anterior subrutina en assemblador:

Tipus	salts que salten	salts que no salten	load/store	les altres
CPI	3	2	10	1

Completa la següent taula indicant, per a cada tipus d'instrucció, el nombre total d'instruccions executades i el nombre total de cicles de rellotge corresponents. Calcula també el temps d'execució total de la subrutina, expressat en cicles i també en nanosegons, tenint en compte que la freqüència de rellotge és de 2GHz.

Tipus	salts que salten	salts que no salten	load/store	les altres
Instruccions	**100**	**1**	**100**	**503**
Cicles	**300**	**2**	**1000**	**503**

Total cicles	**1805**	Total temps (ns)	**902,5**

Pregunta 2. (2,00 punts)

Donat el següent contingut inicial de la memòria representada en hexadecimal a partir de l'adreça 0x10010000, tal i com la mostra el simulador MARS:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	0x040F00FE	0x0043FFE0	0x10010004	0x40000000

a) (1 punt) Sabem que aquest contingut correspon a la següent declaració de variables globals en C, totes elles inicialitzades. Indica les expressions dels requadres **A**, **B0**, **B1**, **C**, **D**, **E** que completin les corresponents inicialitzacions. ATENCIÓ: No s'admet el valor en hexadecimal (es considerarà resposta no vàlida). Tingues en compte que el codi ASCII de la 'A' és 0x41.

```
unsigned char a = [ A ];        /* natural de 8 bits */
short b[2]= { [ B0 ], [ B1 ] };
char c = [ C ];                 /* caràcter ASCII */
int *d = [ D ];
float e = [ E ];
```

A = 254
B0 = 1039
B1 = -32
C = 'C'
D = &b[1]
o bé: D = b+1
E = 2.0

b) (0,5 punts) Quin és el valor final del registre `$t1`, en hexadecimal, després d'executar el següent fragment de codi?

```
li      $t2, 0x10010004
lh      $t3, -2($t2)
lh      $t4, 0($t2)
addu    $t1, $t3, $t4
sh      $t1,  0($t2)
```

`$t1` = 0x **0000 03EF**

Quants bytes de la memòria es modifiquen?

núm. bytes = **2**

Quina variable global es modifica?

b[1]

c) (0,5 punts) Indica el contingut final en hexadecimal del registre `$t1` després d'executar el següent fragment de codi.

```
la      $t0, b
lb      $t1, 2($t0)
andi    $t1, $t1, 0x05FF
```

`$t1` = 0x **0000 05E0**

COGNOMS: **GRUP:**

NOM:

Pregunta 3. (0,75 punts)

El següent codi en MIPS

```
f:      lw      $v0, 380($a0)
        jr      $ra
```

És la traducció de la següent funció incompleta, en C:

```
int f(int matriu[][15])
{
        return matriu[ X ][ Y ];
}
```

Troba les expressions X i Y dels requadres que fan correcta la traducció:

X = 6

Y = 5

Descriu breument els càlculs que has fet per trobar la solució

@matriu[X][Y] = @matriu[0][0] + X·15·4 + Y·4 = $a0 + 380

Per tant: **X·15·4 + Y·4 = 380**

Simplificant, queda: **X·15 + Y = 95**

Però la igualtat ha de satisfer també: Y<15,

Per tant, X és el quocient i Y el residu de dividir 95/15:

X = 95 / 15 = 6, **Y = 95 % 15 = 5**

Pregunta 4. (2,00 punts)

Donades les següents declaracions de funcions en C:

```
int g (char *m, int *n);
char f (int *y)
{
   char V[7];
   int a;
       *(y+1) = g(V, &a);
       return V[a];
}
```

a) (0,50 punts) Segons les regles de l'ABI estudiades, si volem salvar a la pila el mínim nombre de registres durant l'execució de la funció `f`, ¿quines variables i/o paràmetres de la subrutina `f` s'han d'emmagatzemar necessàriament en registres "segurs" `$s` del processador MIPS? Indica també quin registre triaràs en cada cas.

Variable o paràmetre	Registre `$s`
y	**$s0**

b) (0,25 punts) Fes una llista **ordenada** de les variables i registres que componen el **bloc d'activació** de la funció `f` indicant, per a cada un, a quina distància en bytes es troba del cim de la pila (posició on apunta $sp), i quants bytes ocupa. Indica també la mida total del bloc d'activació.

Variable o registre	Distància respecte $sp	Bytes que ocupa
V	**0**	**7**
a	**8**	**4**
$s0	**12**	**4**
$ra	**16**	**4**

Mida total:	**20**

c) (0,75 punts) Tradueix a assemblador la primera sentència de la funció `f`:

```
*(y+1) = g(V, &a);
```

```
move   $s0, $a0          # copia y en registre segur
move   $a0, $sp          # paràmetre V
addiu  $a1, $sp, 8       # paràmetre &a
jal    g
sw     $v0, 4($s0)       # *(y+1) = $v0
```

d) (0,50 punts) Tradueix a assemblador la segona sentència de la funció `f`:

```
return V[a];
```

```
lw     $t0, 8($sp)       # carrega a en $t0
addu   $t0, $sp, $t0     # @V[a] = @V + a*1
lb     $v0, 0($t0)       # carrega V[a] (un char) en $v0
```

COGNOMS: **GRUP:**

NOM:

Pregunta 5. (1,50 punts)

Donat el següent fragment de programa, en C:

```
int M[50][50];
void f(){
        int i;
        for (i=0; i<50; i++)
                M[20][i] = -M[i][20];
}
```

a) (1 punt) Hem traduït el bucle a MIPS, usant la tècnica d'accés seqüencial, amb dos punters:

- El primer punter, emmagatzemat en $t1, apunta a l'element M[i][20] en cada iteració.
- El segon punter, emmagatzemat en $t2, apunta a M[20][i] en cada iteració.

Hem obtingut el següent codi incomplet. Troba les expressions dels requadres **A**, **B**, **C**, **D** perquè la solució sigui correcta.

```
        la      $t1, M + A
        la      $t2, M + B
        li      $t3, 0
        li      $t4, 50
for:
        bge     $t3, $t4, fifor
        lw      $t5, 0 ($t1)
        subu    $t5, $zero, $t5
        sw      $t5, 0 ($t2)
        addiu   $t1, $t1, C
        addiu   $t2, $t2, D
        addiu   $t3, $t3, 1
        b       for
fifor:
```

A = 80

B = 4000

C = 200

D = 4

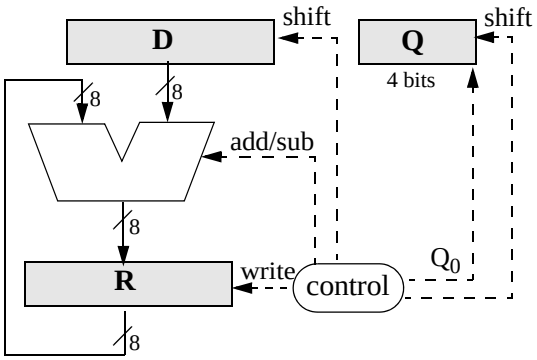
b) (0,50 punts) L'hem optimitzat eliminant la variable d'inducció i convertint-lo a do-while. Les expressions A, B, C, D se suposen les mateixes de l'apartat anterior. Troba les expressions dels requadres **EXPR1** i **EXPR2** perquè sigui correcte.

```
        la      $t1, M + A
        la      $t2, M + B
        la      $t3, EXPR1      # adreça final
do:
        lw      $t5, 0($t1)
        subu    $t5, $zero, $t5
        sw      $t5, 0($t2)
        addiu   $t1, $t1, C
        addiu   $t2, $t2, D
        EXPR2   $t2, $t3, do
```

EXPR1 = M + 4200
EXPR2 = `bltu`
o bé:
EXPR1 = M + 4196
EXPR2 = `bleu`

Pregunta 6. (1,25 punts)

Sigui el següent circuit seqüencial per a la divisió de números naturals de 4 bits, anàleg al que s'ha estudiat durant el curs, el qual calcula el quocient i el residu amb 4 bits:



Suposem que volem calcular la següent divisió (en base 2): 1111/0010

a) (0,50 punts) Quins seran els valors inicials dels tres registres R, D i Q, **en binari**?

R = **0000 1111**

D = **0010 0000**

Q = **0000**

b) (0,25 punts) Quantes sumes i quantes restes ha de fer la ALU per obtenir el resultat final?

Número de sumes = **1**

Número de restes = **4**

c) (0,50 punts) Quins seran els valors finals dels tres registres R, D i Q, **en binari**?

R = **0000 0001**

D = **0000 0010**

Q = **0111**

COGNOMS:	**GRUP:**
NOM:	

Pregunta 7. (1,25 punts)

a) (0,25 punts) Codifica en el format IEE-754 de *simple precisió* el número $x = 1{,}25 \cdot 2^{-126}$, i expressa el resultat en hexadecimal.

$x =$ 0x **00A0 0000**

b) (0,25 punts) Suposem ara que calculem $y = x / 4{,}0$. ¿És possible representar el resultat exacte en el format normalitzat de *simple precisió*? Codifica el resultat exacte en el format *denormal* (no-normalitzat), i expressa'l en hexadecimal.

Es pot expressar *y* en format normalitzat? (S/N) **N**

En format *denormal*: $y =$ 0x **0028 0000**

c) (0,25 punts) Escriu en hexadecimal els següents números en el format de *simple precisió*

El número positiu amb menor valor absolut = 0x **0000 0001**

El número (no infinit) més gran possible = 0x**7F7F FFFF**

d) (0,25 punts) Explica en una sola frase breu: quin avantatge comporta representar l'exponent dels números en coma flotant en el format "en excés" en comptes de "en complement a 2"?

Permet comparar les magnituds amb un comparador de naturals

e) (0,25 punts) Explica en una sola frase breu: a què és degut que la suma de números en coma flotant no tingui la propietat associativa?

És degut a que els arrodoniments introdueixen errors de precisió