

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 ensamblador 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 ensamblador:

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				
Cicles				
Total cicles				Total temps (ns)

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 = ; /* natural de 8 bits */

short b[2] = { , };

char c = ; /* caràcter ASCII */

int *d = ;

float e = .

A =

B0 =

B1 =

C =

D =

E =

- 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 =

Quants bytes de la memòria es modifiquen?

núm. bytes =

Quina variable global es modifica?

- 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 =

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 =

Y =

Describe breument els càlculs que has fet per trobar la solució

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 $\$s$

- 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

Mida total:	
-------------	--

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

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

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

$\text{return } V[a];$

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, ffor
        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
ffor:
```

A =

B =

C =

D =

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.

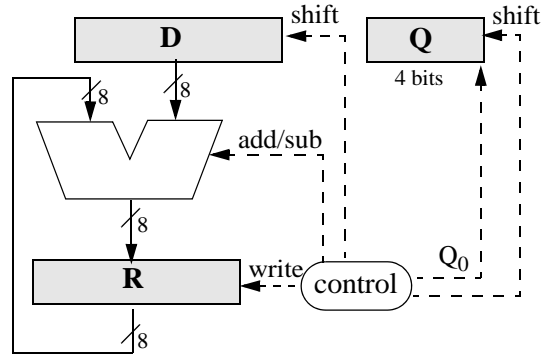
```
        la      $t1, M + A
        la      $t2, M + B
        la      $t3, EXPR1 # adreça final
do:
        lw      $t5, 0($t1)
        subu    $t5, $zero, $t5
        sw      $t5, 0($t2)
        addiu   $t1, $t1, C
        addiu   $t2, $t2, D
        EXPR2 $t2, $t3, do
```

EXPR1 =

EXPR2 =

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 =

D =

Q =

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

Número de sumes =

Número de restes =

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

R =

D =

Q =

COGNOMS:

GRUP:

NOM:

Pregunta 7. (1,25 punts)

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

$x =$

0x

- 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)

En format *denormal*:

$y =$

0x

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

El número positiu amb menor valor absolut =

0x

El número (no infinit) més gran possible =

0x

- 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”?

- 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?