COGNOMS:	**GRUP:**
NOM:	

EXAMEN FINAL D'EC
13 de juny de 2014

L'examen consta de 9 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 duració de l'examen és de 180 minuts. Les notes i la solució es publicaran al Racó el dia 23 de juny. La revisió es farà presencialment el 25 de juny.

Pregunta 1. (1 punt)

Un sistema computador treballa a una freqüència de rellotge de 1 GHz i disposa d'una memòria cache (MC) que utilitza una política d'escriptura immediata sense assignació. Els temps representatius del sistema de memòria són t_h = 1 cicle i t_{block} = 149 cicles.

En aquest computador s'executa un programa de 100 milions d'instruccions, les quals generen 150 milions de referències a memòria. S'ha observat que un 20% de les referències són d'escriptura i que la taxa d'encerts a MC és d'un 90%. El temps total d'execució del programa és 2 s. i la potència dissipada és de 2 W.

a) Determina quin seria el CPI_{ideal} d'aquest sistema computador.

CPI_{ideal} =

b) Si la freqüència del computador s'incrementés a 2 GHz, calcula quin seria el temps d'execució del mateix programa i la potència dissipada.

t_{exe} =

P =

Pregunta 2. (1,20 punts)

Considera un sistema format per un processador amb adreces de 16 bits i una memòria cache (MC) amb la següent configuració:

- capacitat total: 4 blocs
- mida del bloc: 16 bytes
- correspondència directa
- escriptura retardada amb assignació

En un moment donat l'estat de la part de control de la MC és:

índex MC	V	D	etiqueta
0	1	1	1
1	1	1	0
2	1	0	2
3	1	0	3

a) Indica (en hexadecimal) els números de bloc de memòria principal (MP) que hi ha guardats a la MC.

b) Indica una adreça de memòria (en hexadecimal) de la següent referència a tractar que provoqui un encert a la MC.

c) Considera que la propera referència a tractar és una escriptura a l'adreça 0x0190. Indica quin serà l'estat final de la part de control de la MC després de tractar-la.

índex MC	V	D	etiqueta
0			
1			
2			
3			

Quants bytes es transfereixen de MP cap a MC durant la gestió d'aquesta referència?

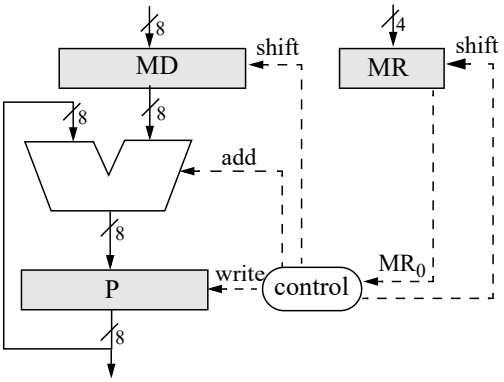
Quants bytes es transfereixen de MC cap a MP durant la gestió d'aquesta referència?

d) Considera ara que la MC és totalment associativa, amb reemplaçament LRU, i amb la mateixa capacitat total i mida de bloc. Sabent que les darreres referències que s'han tractat són: 0x000E, 0x00FE, 0x00AC, 0x000A, 0x0018 i 0x00CE, indica quines són les etiquetes (en hexadecimal) que hi ha guardades a la part de control de la MC.

COGNOMS:	**GRUP:**
NOM:	

Pregunta 3. (0,60 punts)

Sigui el següent diagrama del multiplicador seqüencial de nombres naturals de 4 bits anàleg al que s'ha estudiat durant el curs, el qual calcula el producte amb 8 bits:



Suposem que amb aquest circuit multipliquem els números binaris de 4 bits 1001 (multiplicand) i 1011 (multiplicador). Completa la següent taula, que mostra els valors en binari dels registres P, MD, i MR després de la inicialització i després de cada iteració, afegint tantes iteracions com facin falta:

iter.	P (Producte)								MD (Multiplicand)								MR (Multiplicador)			
ini	0	0	0	0	0	0	0	0									1	0	1	1
1																				
2																				

Pregunta 4. (1,50 punts)

Donada la següent declaració de funcions en C:

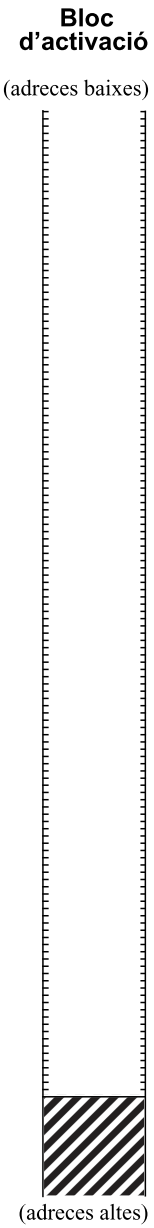
```
void subr2(short *x, int *y, char *z);
void subr1(int a[], int b) {
  char k;
  short v[7];
  int i;
        for (i = 0; i < b; i++)
                subr2(v, &a[3], &k);
}
```

Quins elements de `subr1` (paràmetres, variables locals i càlculs intermedis) s'han de guardar en registres de tipus segur **$s**?

Dibuixa el bloc d'activació de `subr1` (sobre el diagrama que tens a la dreta), especificant-hi la posició on apunta el registre `$sp` un cop reservat l'espai corresponent a la pila, així com el nom de cada registre i/o variable, i la seva posició (desplaçament relatiu al `$sp`).

Tradueix a MIPS la subrutina `subr1`.

```
subr1:
```

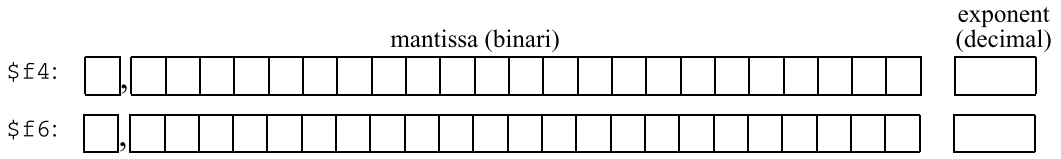


COGNOMS:	**GRUP:**
NOM:	

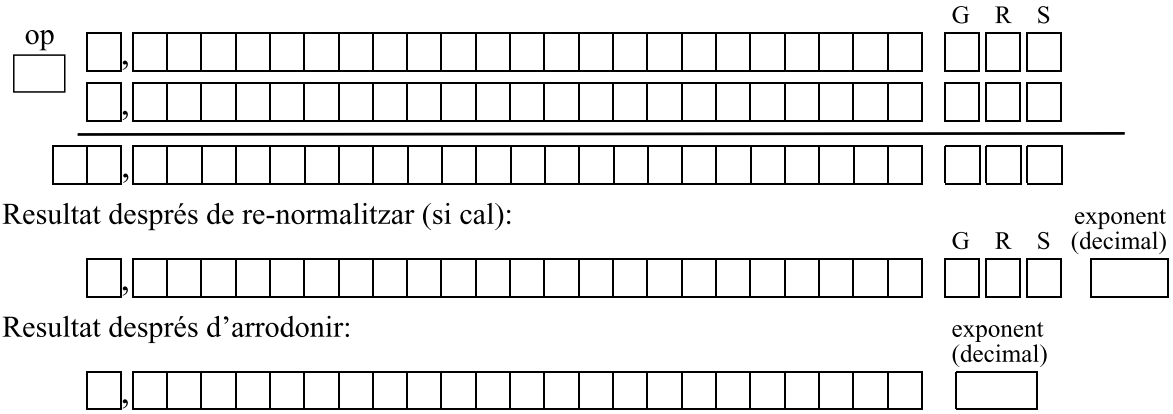
Pregunta 5. (1,20 punts)

Considera que el contingut dels registres `$f4` i `$f6` és **0x40800003** i **0xBF700005**, respectivament i que s'executa la instrucció MIPS: **`add.s $f0,$f4,$f6`**. Suposant que el sumador/restador té 1 bit de guarda, un d'arrodoniment i un de "sticky", i que arrodoneix al més pròxim (al parell en el cas equidistant), contesta les següents preguntes:

Quina és la mantissa (en binari) i l'exponent (en decimal) dels nombres que hi ha a `$f4` i `$f6`?



Omple les següents caselles mostrant l'operació op (+/-), les cadenes de bits a operar, i el resultat:



Quin és el valor de `$f0` en hexadecimal després d'executar la instrucció ?

`$f0 =` `0x`

COGNOMS:	**GRUP:**
NOM:	

Pregunta 7. (1 punt)

Posa una X al costat de cada una de les següents afirmacions (a la columna V si és Verdadera o a la columna F si és Falsa). Suposem en tots els casos que es fa referència a un processador MIPS com l'estudiat a classe. Cada resposta correcta suma 0,1 punts; les respostes no contestades no es tenen en compte; cada resposta incorrecta resta 0,1 punts; i la puntuació total mínima és 0.

	Afirmació	V	F
1.-	La rutina RSE coneix la causa (codificada) de l'excepció perquè aquesta se li passa com a paràmetre en `$a0` (paràmetre `ExcCode`).		
2.-	En un processador amb adreces de 32 bits, una cache associativa de 2 vies, de 256KB i blocs de 16 bytes, s'han de dedicar 15 bits a etiqueta (TAG), 13 a número d'entrada (conjunt) i 4 a desplaçament.		
3.-	Dividir per 2 un número enter en complement a 2 que estigui a `$t0` és sempre equivalent a fer: `sll $t0,$t0,1`		
4.-	Es pot produir més d'una excepció per fallada de TLB en l'execució d'una instrucció.		
5.-	En les memòries cache que utilitzen escriptura immediata s'ha de posar el *dirty bit* a 1 només quan hi ha un encert d'escriptura.		
6.-	El temps de penalització en cas de fallada d'escriptura d'una memòria cache d'escriptura immediata sense assignació és zero segons el model estudiat.		
7.-	Si s'executa un mateix programa en dues memòries cache diferents, però les dues de correspondència directa, sempre tindrà una major taxa d'encerts la que tingui els blocs de mida més gran.		
8.-	Un sistema que tingués una memòria cache més gran que la memòria principal mai tindria fallades.		
9.-	Un processador sense TLB pot suportar memòria virtual.		
10.-	Si una subrutina `f` està declarada com `void f(float *p)` el paràmetre `p` s'ha de passar en el registre `$f12`.		

Pregunta 8. (1 punt)

Donat el següent codi en alt nivell:

```
int M[10][10];

main() {
int i,j;
       j=4;
       for (i=0; i<5; i++) {
              M[i][j] = M[4][i+j];
       }
}
```

Completa el següent codi en assemblador del MIPS, que fa el mateix que el programa donat, però que utilitza la tècnica d'accés seqüencial, on `$t0` és el punter als elements `M[i][j]`, `$t1` és el punter als elements `M[4][i+j]` i `$t3` conté l'adreça de la matriu tal que no hi ha d'arribar el punter `$t0` i, per tant, serveix per controlar la finalització del bucle.

```
main:
    la     $t0,  [          ]  # punter a M[i][j]

    la     $t1,  [          ]  # punter a M[4][i+j]

    la     $t3,  [          ]  # adreça límit punter $t0
bucle:
    bgeu $t0,$t3, fibucle
    lw $t2,0($t1)
    sw $t2,0($t0)
    addiu $t0,$t0,  [          ]

    addiu $t1,$t1,  [          ]

    b bucle
fibucle:
    jr $ra
```

COGNOMS:	GRUP:
NOM:	

Pregunta 9. (1 punt)

Donades les següents declaracions de variables globals, emmagatzemades a memòria a partir de l'adreça 0x10010000:

```
v1:.word 0x0100100C, 0x10010010
v2:.half 5,-2, 0x1234
v3:.byte 1
v4:.dword 0x1111
v5:.word 0x89AB
v6:.dword 0xAE8
```

a) Omple la següent taula amb el contingut de memòria en hexadecimal. Les posicions de memòria sense inicialitzar s'han de posar a 0.

@Memòria	Dada
0x10010000	
0x10010001	
0x10010002	
0x10010003	
0x10010004	
0x10010005	
0x10010006	
0x10010007	
0x10010008	
0x10010009	
0x1001000A	
0x1001000B	
0x1001000C	
0x1001000D	
0x1001000E	
0x1001000F	

@Memòria	Dada
0x10010010	
0x10010011	
0x10010012	
0x10010013	
0x10010014	
0x10010015	
0x10010016	
0x10010017	
0x10010018	
0x10010019	
0x1001001A	
0x1001001B	
0x1001001C	
0x1001001D	
0x1001001E	
0x1001001F	

@Memòria	Dada
0x10010020	
0x10010021	
0x10010022	
0x10010023	
0x10010024	
0x10010025	
0x10010026	
0x10010027	
0x10010028	
0x10010029	
0x1001002A	
0x1001002B	
0x1001002C	
0x1001002D	
0x1001002E	
0x1001002F	

@Memòria	Dada
0x10010030	
0x10010031	
0x10010032	
0x10010033	
0x10010034	
0x10010035	
0x10010036	
0x10010037	
0x10010038	
0x10010039	
0x1001003A	
0x1001003B	
0x1001003C	
0x1001003D	
0x1001003E	
0x1001003F	

b) Quin és el valor final del registre `$t2`, en hexadecimal, després d'executar el següent fragment de codi?

```
la     $t0, v1+4
lw     $t1, 0($t0)
lw     $t2, 8($t1)
lb     $t3, 16($t1)
xor    $t2, $t3, $t2
```

`$t2 =` 0x