COGNOMS:	**GRUP:**
NOM:	

EXAMEN FINAL D'EC
17 de gener de 2023

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 gener. La revisió es farà presencialment el 24 de gener, de 8 a 9 (preguntes 1 a 4) i de 9 a 10h (preguntes 5 a 9).

Pregunta 1. (1 punt)

Un processador que funciona a una freqüència de rellotge de 5Ghz disposa de 3 tipus d'instruccions diferents: A, B i C. La següent taula mostra quin és el nombre d'instruccions executades de cada tipus durant l'execució d'un programa de test, el CPI promig de cada tipus d'instrucció, i la potència dissipada durant l'execució de cada tipus d'instrucció, en Watts (Joules/segon).

Tipus d'instrucció	Nombre d'instruccions	CPI	Potència (W)
A	$9 \cdot 10^{10}$	2	10
B	$5 \cdot 10^{10}$	10	4
C	$4 \cdot 10^{10}$	3	5

a) Calcula el temps total d'execució del programa, en segons

t_{exe} = [ ] s

b) Calcula el consum total d'energia del programa, en Joules

E = [ ] J

c) Al redissenyar el processador, la freqüència del rellotge es redueix a 4GHz i les instruccions de tipus B passen a tenir només 6 cicles per instrucció. Calcula el guany de rendiment (speed-up) obtingut després del redisseny, donant el resultat amb 2 xifres decimals de precisió.

Guany = [ ]

Pregunta 2. (1,25 punts)

Donada la següent declaració de variables globals d'un programa escrit en llenguatge C:

```
short a[] = {255, 11, -9};
char b[] = "2023";   /* s'usen tants bytes com calgui pel string */
short c = 0;
long long d = 18;
short *e = &c;
```

a) Tradueix-la al llenguatge assemblador del MIPS

```
        .data
```

b) Completa la següent taula amb el contingut de memòria en hexadecimal (sense el prefix "0x") de les primeres 32 posicions de memòria. Tingues en compte que el codi ASCII del '0' és el 0x30. Les variables s'emmagatzemen a partir de l'adreça 0x10010000. Les posicions de memòria no ocupades es deixen en blanc.

@Memòria	Dada
0x10010000	
0x10010001	
0x10010002	
0x10010003	
0x10010004	
0x10010005	
0x10010006	
0x10010007	

@Memòria	Dada
0x10010008	
0x10010009	
0x1001000A	
0x1001000B	
0x1001000C	
0x1001000D	
0x1001000E	
0x1001000F	

@Memòria	Dada
0x10010010	
0x10010011	
0x10010012	
0x10010013	
0x10010014	
0x10010015	
0x10010016	
0x10010017	

@Memòria	Dada
0x10010018	
0x10010019	
0x1001001A	
0x1001001B	
0x1001001C	
0x1001001D	
0x1001001E	
0x1001001F	

c) Quin és el valor final de $t0 i $t1, en hexadecimal, després d'executar aquest fragment?

```
la      $t0, b
lbu     $t0, 3($t0)
addiu   $t0, $t0, '1'
sltu    $t1, $zero, $t0
```

$t0 = 0x

$t1 = 0x

d) Tradueix a llenguatge assemblador del MIPS la següent sentència en C:

```
*e = a[0] - a[2];
```

COGNOMS:	**GRUP:**
NOM:	

Pregunta 3. (1 punt)

Donada la següent funció en C:

```
int f (int i, int M[][400]) {
        return M[i+2][i-4];
}
```

Completa els requadres del següent fragment de codi en assemblador MIPS per tal que sigui la traducció correcta de la funció anterior:

```
f:  li      $t0, [          ]
    mult    $t0, [          ]
    mflo    $t0
    addu    $t0, $t0, [          ]
    lw      $v0, [          ]($t0)
    jr      $ra
```

Pregunta 4. (1,50 punts)

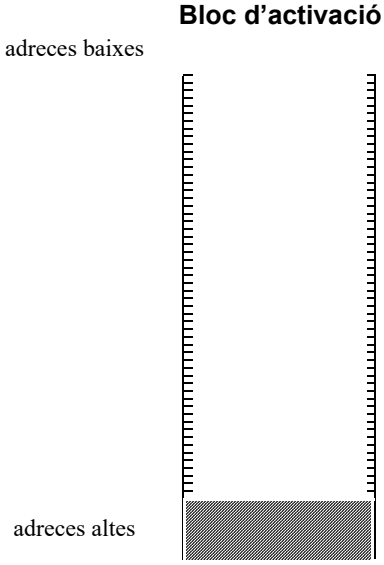
Donat el següent codi en C:

```
int *sub1(int *a, int b);
int sub2(int c, int *d);

int sub3(int *e, int f, int i) {
   int *g, h=0;
   while (h < f) {
          g = sub1(e, i);
          h = sub2(*g, &h);
   }
   return h;
}
```

a) Seguint les regles de l'ABI estudiades, determina quines variables, paràmetres o càlculs intermedis de la funció `sub3` cal guardar obligatòriament en registres segurs per garantir que no siguin alterats per les crides a `sub1` o `sub2`.

b) Dibuixa el bloc d'activació de `sub3`, especificant el nom i posició (relatius al $sp) de cada element que conté, així com la posició on apunta $sp un cop creat el bloc d'activació.



c) Tradueix a MIPS la subrutina `sub3`

COGNOMS:	GRUP:
NOM:	

Pregunta 5. (1 punt)

a) Considera que treballem amb nombres enters representats en complement a 2 i en 16 bits. Indica en hexadecimal quin és el nombre *x* més petit tal que la resta 0x7FFF – *x* genera sobreeiximent (o *overfow*).

x = 0x

b) Considera que compilem el següent programa en C per al processador MIPS

```
main( ) {
short a = 0x789A;
short b = -16;
short c, d, e;
          c = a / b;
          d = a % b;
          e = a >> 4;
}
```

Indica en hexadecimal (amb 4 dígits) quin és el contingut de les variables *c*, *d*, *e* després d'executar l'anterior programa.

c = 0x d = 0x e = 0x

Pregunta 6. (1 punt)

Suposem que tenim un processador de 32 bits amb una memòria cache de dades de 256 bytes, on cada bloc té 16 bytes. Suposem que executem el següent programa.

```
int M[4][32];

void main() {
int i, j;                          // i, j en registre
    for (i=0; i<3; i++)
      for (j=0; j<32; j++)
         M[i][j] = M[i+1][j];
}
```

Calcula el nombre de fallades de la cache suposant que la memòria cache és inicialment buida. L'adreça base de la matriu M és 0.

a) Suposant que la cache és de correspondència directa i té la política d'**escriptura retardada amb assignació**.

fallades =

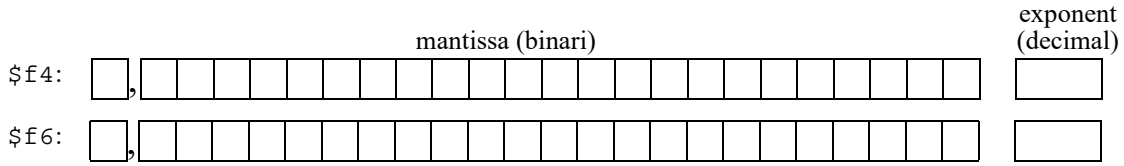
b) Suposant que la cache és **associativa per conjunts de 4 vies** (algorisme de reemplaçament LRU), i que té la política d'**escriptura immediata sense assignació**.

fallades =

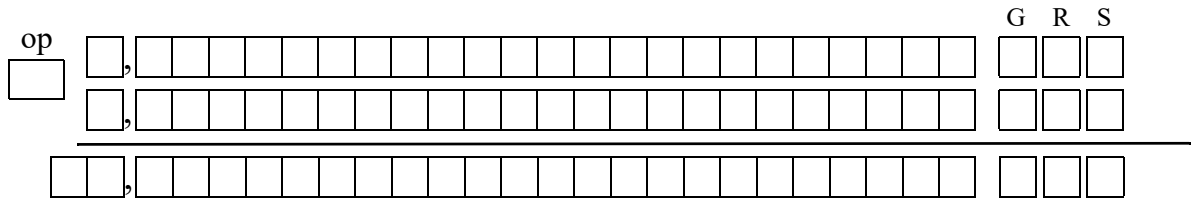
Pregunta 7. (1 punt)

Considera que el contingut dels registres `$f4` i `$f6` és `$f4` = **0x40000031** i `$f6` = **0x4200000D**, i s'executa la instrucció MIPS: **`sub.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`?



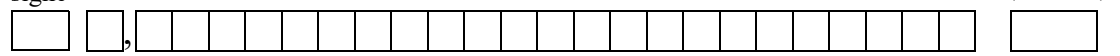
Omple les caselles mostrant l'operació op (+/-), els bits a operar, i el resultat en valor absolut:



Resultat en valor absolut després de normalitzar (si cal):

																								G	R	S	exponent (decimal)
,	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	0	4

Resultat amb signe (+/-) després d'arrodonir:



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

`$f0 =` `0x` _____

COGNOMS:	**GRUP:**
NOM:	

Pregunta 8. (1,25 punts)

La memòria virtual implementada en un sistema computador de 32 bits es caracteritza pels següents paràmetres:

- Pàgines de 256 bytes de mida.
- Un màxim de 5 pàgines carregades simultàniament a memòria física per aplicació.
- Reemplaçament de pàgines a memòria física seguint l'algorisme LRU.
- TLB totalment associatiu de 4 entrades amb reemplaçament LRU.

Donat el següent programa en C:

```
int V[384];
void main() {
        int i;
        for (i=0; i < 384; i++) {
                V[383 - i] = V[i];
        }
}
```

Considera que la variable local *i* s'emmagatzema en un registre, que el vector global V s'emmagatzema en memòria a partir de l'adreça `0x00000000`, i que el codi s'emmagatzema a partir de l'adreça `0x00000A00`, i ocupa menys d'una pàgina. El TLB i la memòria física estan inicialment buits. Es demana:

a) Quantes pàgines ocupa el vector `V`?

nombre de pàgines = ____

b) Quantes fallades de TLB (codi i dades) es produiran en tota l'execució del programa?

fallades de TLB = ____

c) Quantes fallades de pàgina (codi i dades) es produiran en tota l'execució del programa?

fallades de pàgina = ____

d) Indica els VPN (en hexadecimal) de les cinc pàgines (codi i/o dades) que hi haurà carregades a memòria física quan s'acabi l'execució del programa.

VPNs = ____

Pregunta 9. (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.-	En format de simple precisió IEEE-754 (32 bits), la codificació 0x00F00000 representa un número *denormal* (no normalitzat).		
2.-	La suma de números en coma flotant té les propietats commutativa i associativa.		
3.-	La codificació en excés de l'exponent dels nombres en coma flotant permet comparar les seves magnituds amb un simple comparador de naturals.		
4.-	En un sistema amb memòria virtual, la mida total d'un programa i les seves dades poden excedir la capacitat de la memòria física.		
5.-	Si el bit EXL val 1, les interrupcions seran ignorades.		
6.-	Durant el processament de l'excepció causada per una instrucció `syscall`, la RSE li suma 4 al registre EPC abans de retornar al programa.		
7.-	La rutina RSE de tractament d'excepcions del MIPS segueix les mateixes regles de l'ABI que s'estableixen per programar les subrutines.		
8.-	La rutina de tractament de fallades de TLB està optimitzada de manera que no li cal accedir a la Taula de Pàgines.		
9.-	Quan la CPU detecta una excepció, s'interromp la instrucció en curs per donar pas al tractament de l'excepció.		
10.-	Quan la CPU rep una petició d'interrupció, s'interromp la instrucció en curs per donar pas al tractament de la interrupció.		