COGNOMS:	**GRUP:**
NOM:	

EXAMEN FINAL D'EC
9 de gener de 2020

L'examen consta de 10 preguntes, que s'han de contestar als mateixos fulls de l'enunciat. Posa el teu nom i cognoms a tots els fulls. La duració de l'examen és de 180 minuts. Les notes es publicaran al Racó el dia 20 de gener. La revisió es farà presencialment el 21 de gener de 11 a 12h.

Pregunta 1. (0,9 punts)

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.-	Per obtenir el resultat en n bits de la multiplicació de dos nombres enters negatius representats en Ca2 només cal fer el seu producte amb un multiplicador de nombres naturals, sense fer cap tractament de signes a priori ni a posteriori.	**X**	
2.-	Si es permetés que l'RSE pogués tractar excepcions llavors el sistema entraria en un bucle infinit en el moment en què es produís una excepció mentre s'executa l'RSE.		**X**
3.-	La divisió entera per una potència de 2 és sempre equivalent a un desplaçament aritmètic a la dreta de tants bits com indiqui l'exponent de la potència.		**X**
4.-	L'excepció per accés no alineat a memòria pot ser inhibida a través del camp Interrupt Mask.		**X**
5.-	Una excepció no pot ser atesa fins que la instrucció en curs hagi finalitzat.		**X**
6.-	Al format de coma flotant IEEE 754 de simple precisió, l'exponent 255 representa tant infinit, com menys infinit com el concepte NaN (Not a Number).	**X**	
7.-	En un sistema amb memòria virtual que té una mida de pàgina fixa, la mida de la taula de pàgines augmentarà tant si s'augmenta la mida de l'espai d'adreçament virtual com si s'augmenta la mida de l'espai d'adreçament físic.	**X**	
8.-	Per a una fallada de lectura, una cache d'escriptura immediata sense assignació té el mateix temps de penalització que una cache d'escriptura retardada amb assignació.		**X**
9.-	Un accés a memòria mai canviarà l'estat de la taula de pàgines si es produeix un encert al TLB a una entrada amb el bit de validesa a 1.		**X**

Pregunta 2. (1,1 punts)

Tenim la instrucció **`div.s $f0, $f2, $f4`**
Sabem que després d'executar la instrucció `$f0` = 0x40100001.
També sabem que `$f4` = 0x3FC00000.

Calcula un valor (en hexadecimal) de `$f2` que compleixi aquestes condicions.

`$f2` = **0x40580002 (també 0x40580001)**

Pregunta 3. (1 punt)

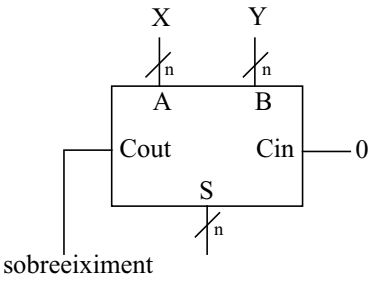
Per a cadascun dels apartats següents, on S, X i Y són nombres d'n bits, indica textualment i de forma concisa com es pot saber si la suma S = X+Y produeix sobreeiximent i dissenya, també, un circuit que ho implementi.

NOTA: Pots usar tots els blocs combinacionals i les portes que creguis convenient. En cas d'usar un sumador aritmètic, considera que les seves entrades són A i B d'n bits i Cin d'1 bit i té les sortides S d'n bits i Cout d'1 bit.

a) [0,5 punts] S, X i Y són nombres naturals

i) Hi ha transport en la suma aritmètica d'X i Y en n bits

Altres possibilitats:
ii) Sent S = X+Y, si S < X (o també si S < Y)
iii) Si X > $\overline{Y}$ (o també si Y>$\overline{X}$)



b) [0,5 punts] S, X i Y són nombres enters representats en Ca2

i) Es sumen dos nombres del mateix signe i el signe del resultat és oposat.

A partir del mateix circuit anterior caldria agafar els bits de més pes d'X, Y i S (X_{n-1}, Y_{n-1} i S_{n-1}, respectivament) i fer:

$$\text{sobreeiximent} = X_{n-1} * Y_{n-1} * \overline{S_{n-1}} + \overline{X_{n-1}} * \overline{Y_{n-1}} * S_{n-1}$$

Altres possibilitats (sent C_{n-1} el transport que hi ha d'entrada a la suma dels bits n-1):
ii) Cout * (X_{n-1} XOR Y_{n-1})
iii) Cout XOR C_{n-1}
iv) X_{n-1} XOR Y_{n-1} XOR C_{n-1}

COGNOMS:	GRUP:
NOM:	

Pregunta 4. (0,7 punts)

Donada la següent funció en C:

```
short func(short M[][1000], int i) {
       return M[i][i+3] + M[i+2][i];
}
```

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

```
unc:    li      $t0,  2002
        mult    $t0,  $a1
        mflo    $t0
        addu    $t0, $t0,  $a0
        lh      $t1,  6     ($t0)
        lh      $t2,  4000  ($t0)
        addu    $v0, $t1, $t2
        jr      $ra
```

Pregunta 5. (1,1 punts)

Un sistema computador treballa a una freqüència de rellotge de 1 GHz, un voltatge de 1 V i disposa d'una memòria cache (MC) que utilitza una política d'escriptura **immediata amb assignació**. Els temps representatius del sistema de memòria són $t_h = 1$ cicle i $t_{block} = 199$ cicles. En aquest computador s'executa un programa de $3 \cdot 10^9$ instruccions, les quals generen $2 \cdot 10^9$ referències a memòria. S'ha observat que la taxa d'encerts a MC és d'un 90%. El temps total d'execució del programa és 52 segons i la potència dissipada és de 2 W.

a) [0,5 punts] Determina quin seria el CPI_{ideal} d'aquest sistema computador.

CPI_{ideal} = **4**

b) [0,6 punts] Si la freqüència del computador s'incrementés a 2 GHz i el voltatge s'incrementés a 1.2 V, suposant que no ha canviat la capacitància (C) ni el factor d'activitat (α), ni el CPI_{ideal} ni cap paràmetre del sistema de memoria, calcula quin seria el temps d'execució del mateix programa i la potència dissipada.

t_{exe} = **26 s**

P = **5.76 W**

Pregunta 6. (1 punt)

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

```
char a = 'A';
float b = 1.5;
short c[3] = {-2, -1, 7};
float *d = &b;
long long e[100];
```

a) [0,3 punts] Tradueix-la al llenguatge assemblador del MIPS.

```
        .data
a:      .byte 'A'
b:      .float 1.5
c:      .half -2, -1, 7
d:      .word b
        .align 3
e:      .space 800
```

b) [0,3 punts] Completa la següent taula amb el contingut de memòria en hexadecimal. Tingues en compte que el codi ASCII de la 'A' és el 0x41. Les variables s'emmagatzemen a partir de l'adreça 0x10010000. Les posicions de memòria sense inicialitzar es deixen en blanc.

@Memòria	Dada
0x10010000	**41**
0x10010001	
0x10010002	
0x10010003	
0x10010004	**00**
0x10010005	**00**
0x10010006	**C0**
0x10010007	**3F**

@Memòria	Dada
0x10010008	**FE**
0x10010009	**FF**
0x1001000A	**FF**
0x1001000B	**FF**
0x1001000C	**07**
0x1001000D	**00**
0x1001000E	
0x1001000F	

@Memòria	Dada
0x10010010	**04**
0x10010011	**00**
0x10010012	**01**
0x10010013	**10**
0x10010014	
0x10010015	
0x10010016	
0x10010017	

@Memòria	Dada
0x10010018	**00**
0x10010019	**00**
0x1001001A	**00**
0x1001001B	**00**
0x1001001C	**00**
0x1001001D	**00**
0x1001001E	**00**
0x1001001F	**00**

c) [0,4 punts] Donat el següent codi en assemblador MIPS, indica quin és el valor final en hexadecimal del registre `$t0`:

```
la      $t0, c
lh      $t1, 2($t0)
lh      $t2, 4($t0)
sll     $t2, $t2, 4
xor     $t0, $t1, $t2
```

$t0 = **0xFFFFFF8F**

COGNOMS:	GRUP:
NOM:	

Pregunta 7. (0,5 punts)

Donada la següent sentència escrita en alt nivell en C:

```
if ((x > y) && (y < 0))
    x++;
else if ((x <= 0) || (y <= 0))
    y++;
```

Completa el següent fragment de codi en MIPS, que tradueix l'anterior sentència, escrivint en cada calaix un mnemònic d'instrucció o macro, etiqueta, registre o immediat. Les variables `x` i `y` són de tipus `int` i estàn inicialitzades i guardades als registres `$t0` i `$t1` respectivament.

```
        ble     $t0, $t1,    etq4
etq1:   bge     $t1, $zero,  etq4
etq2:   addiu   $t0, $t0, 1
etq3:   b       etq7
etq4:   ble     $t0, $zero,  etq6
etq5:   bgt     $t1, $zero,  etq7
etq6:   addiu   $t1, $t1, 1
etq7:
```

Pregunta 8. (1,1 punts)

Donades les següents declaracions de funcions en C:

```
float g(int *v, char *b);

float f(float *v) {
        int w[10];
        char a;
        return g(&w[2], &a) + v[2];
}
```

Tradueix a MIPS la funció f:

```
f:  addiu  $sp, $sp, -52
    sw     $s0, 44($sp)
    sw     $ra, 48($sp)
    move   $s0, $a0
    addiu  $a0, $sp, 8
    addiu  $a1, $sp, 40
    jal    g
    lwc1   $f2, 8($s0)
    add.s  $f0, $f0, $f2
    lw     $s0, 44($sp)
    lw     $ra, 48($sp)
    addiu  $sp, $sp, 52
    jr     $ra
```

COGNOMS:	GRUP:
NOM:	

Pregunta 9. (1,3 punts)

Tenim un programa que, donada una matriu `M`, calcula la seva matriu trasposada `T`. El codi del programa és el següent:

```
int M[32][64], T[64][32];

for (int i=0; i<64; i++)
        for (int j=0; j<32; j++)
                T[i][j] = M[j][i];
```

Al compilar el programa, el compilador emmagatzema la matriu `M` a l'adreça 0x10010000 i la matriu `T` immediatament a continuació, mentre que les variables `i` i `j` s'emmagatzemen a registres del processador.

El programa s'executa a un processador de 32 bits amb una memòria cache amb les següents característiques:

- Capacitat de 64 blocs de 32 Bytes cadascun
- Completament associativa amb algorisme de reemplaçament LRU
- Escriptura retardada amb assignació

Es demana:

a) [0,8 punts] Quantes fallades de cache es produiran durant l'execució del programa?. Pots argumentar la resposta.

fallades = **512**

Els accessos a la matriu T causen 4 fallades a cada iteració del bucle 'i'. Els accessos a la matriu M causen 32 fallades cada 8 iteracions del bucle 'i'. En total tenim 4*64+32*64/8=512 fallades.

b) [0,5 punts] Quantes fallades de cache es produiran durant l'execució del programa si només es canvia la política d'escriptura de la cache a escriptura immediata sense assignació?

fallades = **2304**

Pregunta 10. (1,3 punts)

Tenim un processador amb memòria virtual basada en paginació amb les següents característiques:

- Adresses lògiques de 20 bits
- Pàgines de 16KB

A continuació es mostra l'estat inicial de la taula de pàgines d'un programa en execució. L'ordre temporal en que s'ha accedit a les pàgines virtuals és 2,4,1 (la pàgina 2 és la més antiga). :

VPN	P	D	PPN
0	0	0	
1	1	0	2
2	1	1	0
3	0	0	
4	1	0	1
5	0	0	
6	0	0	
7	0	0	
...			
63	0	0	

El programa pot tenir a memòria física un màxim de 4 pàgines simultàniament que es mapegen sempre als marcs de pàgina física 0, 1, 2 i 3. Tal com mostra la taula de pàgines, a l'estat inicial la memòria física té 3 pàgines virtuals carregades als marcs de pàgina física 0, 1 i 2, i el marc de pàgina física 3 està inicialment lliure. En cas que una pàgina s'hagi de sacrificar per deixar lloc a una altra, es fa servir l'algorisme de reemplaçament LRU.

A partir d'aquest estat inicial s'executen una seqüència de sis accessos a memòria. Emplena la següent taula indicant, per a cada accés a memòria, el seu número de pàgina virtual (VPN), si hi ha fallada de pàgina, si es llegeix la pàgina de disc, quina pàgina virtual es reemplaça, quina pàgina virtual s'escriu a disc (si cal), i quin és el marc de pàgina física (PPN) resultant de la traducció.

L/E	Adr. lògica (hex)	VPN (hex)	Fallada de pàgina? (SI/NO)	Lectura de disc (SI/NO)	VPN reempl. (hex)	VPN escrita a disc (hex)	PPN (hex)
L	0x1C124	**7**	**SI**	**SI**	-	-	**3**
E	0x12ACB	**4**	**NO**	**NO**	-	-	**1**
L	0x0D972	**3**	**SI**	**SI**	**2**	**2**	**0**
L	0x08448	**2**	**SI**	**SI**	**1**	-	**2**
E	0x1E666	**7**	**NO**	**NO**	-	-	**3**
L	0x07A34	**1**	**SI**	**SI**	**4**	**4**	**1**