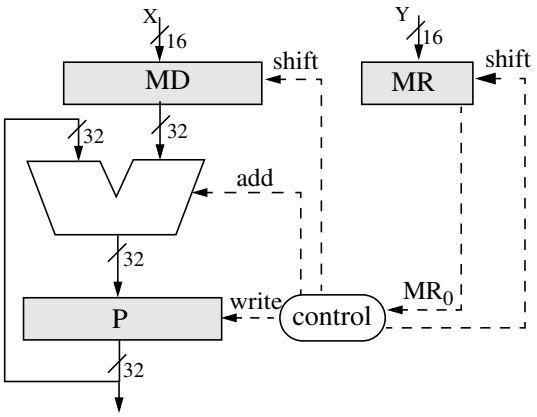
COGNOMS:	**GRUP:**
NOM:	

EXAMEN FINAL D'EC
23 de gener de 2018

L'examen consta de 10 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, la solució i el procediment de revisió es publicaran al Racó el dia 29 de gener. La revisió es farà el 30 de gener.

Pregunta 1. (0,90 punts)

Sigui el circuit seqüencial per a la multiplicació de números naturals de 16 bits, anàleg a l'estudiat a classe, el qual calcula el producte en 32 bits:



Escriu usant el llenguatge C l'algorisme que descriu les operacions que ha de realitzar la unitat de control (anàleg al pseudocodi explicat a classe). És a dir, programa la següent funció `multu` que té els registres `MD`, `MR` i `P` com a variables locals, i els operands `X`, `Y` com a paràmetres, i on el producte és el resultat a retornar:

```
unsigned int multu(unsigned short X, unsigned short Y) {
unsigned int MD, P;
unsigned short MR;









}
```

Pregunta 2. (1,40 punts)

Considera la següent funció en llenguatge d'alt nivell:

```
int fib(int n) {
int tmp;
        if (n<2)
                tmp = n;
        else
                tmp = fib(n-1) + fib(n-2);
        return tmp;
}
```

Contesta els següents apartats suposant que volem fer servir el mínim nombre indispensable de registres segurs en la programació d'aquesta funció com una subrutina:

El paràmetre `n` cal salvar-lo en un registre segur? Per què?

La variable local `tmp` s'ha de guardar en un registre segur o en un de temporal? Per què?

Enumera tots els registres segurs (només els `$s`) que faràs servir en la subrutina indicant per cadascun d'ells quina és la dada que emmagatzema.

Quants bytes ocupa el bloc d'activació de la subrutina?

Programa en llenguatge MIPS la subrutina corresponent a la funció `fib`.

```
fib:
```

COGNOMS:	**GRUP:**
NOM:	

Pregunta 3. (0,80 punts)

Donada la següent funció en C:

```
short acces_aleatori(short M[][100], int i) {
        return M[i+2][i-1];
}
```

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

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

Pregunta 4. (1,10 punts)

Considerem un computador amb un processador MIPS funcionant a una freqüència de 500Mhz, i que dissipa una potència de 20 W. Suposem que la cache d'instruccions és ideal (sempre encerta), i que la cache de dades té un temps de servei en cas d'encert t_h = 1 cicle. El temps necessari per copiar un bloc de memòria principal a cache és t_{block} = 59 cicles. Els CPI dels diversos tipus d'instruccions (en absència de fallades) és:

	Salts	Loads	Resta d'instruccions
CPI	8	3	1

A través de simulacions amb un programa de test hem mesurat una taxa de fallades de cache de m = 2,5%. Totes les referències a memòria són lectures. El nombre d'instruccions executades és:

	Salts	Loads	Resta d'instruccions
n. instr.	$3 \cdot 10^9$	$6 \cdot 10^9$	$20 \cdot 10^9$

a) (0,5 pts) Calcula el temps d'accés mitjà a memòria dels loads per a aquest programa, en cicles

t_{am} = [ ] cicles

b) (0,5 pts) Calcula el temps d'execució del programa (incloent-hi fallades de cache), en segons

t_{exe} = [ ] s

c) (0,1 pts) Calcula l'energia total consumida durant l'execució del programa, en Joules

E = [ ] J

Pregunta 5. (0,50 punts)

Donada la següent funció en C:

```
void func(unsigned short *p) {
        *(p + 5) = *p + 5;
}
```

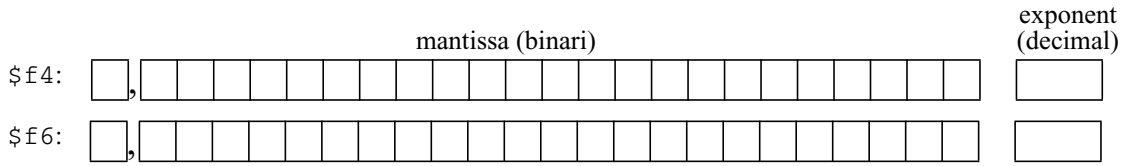
Tradueix a MIPS la funció `func` amb el mínim nombre de línies de codi:

```
func:
```

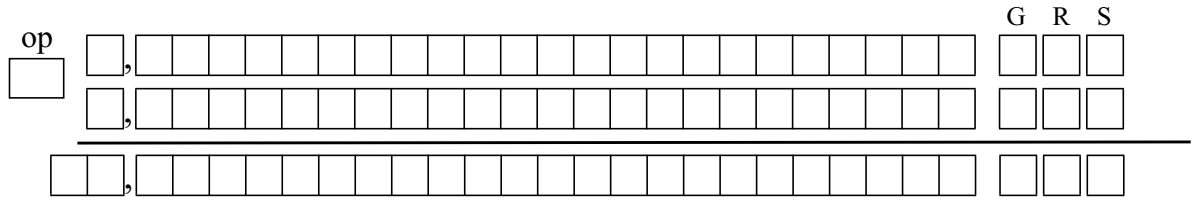
Pregunta 6. (1,30 punts)

Considera que el contingut dels registres `$f4` i `$f6` és **0xC0800003** i **0x3EA00009** 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:

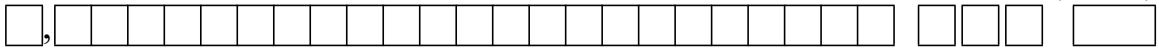
Quina és la mantissa (en binari) i l'exponent (en decimal) dels nombres que hi ha a `$f4` i `$f6`?



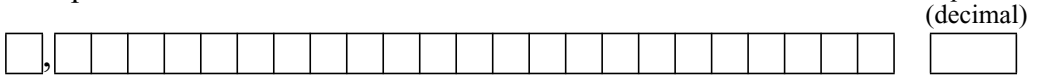
Omple les següents caselles mostrant l'operació op (+/-), les cadenes de bits a operar, i el resultat:



Resultat després de re-normalitzar (si cal):



Resultat després d'arrodonir:



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

`$f0 =` `0x`

COGNOMS:	**GRUP:**
NOM:	

Pregunta 7. (0,60 punts)

El següent codi C calcula la suma dels elements de la matriu triangular inferior d'`M`:

```
/* versió A */
int M[5][5];
int suma=0;

main() {
int i,j;

       for (i=0; i<5; i++)
           for (j=0; j<=i; j++)
              suma += M[i][j];
}
```

Una versió equivalent de l'anterior programa que té 1 sol bucle (inicialitzant el vector `stride` convenientment) és:

```
/* versió B */
int M[5][5];
int suma = 0;
int stride[15] = { ... };

main() {
int i;
int *p = &M[0][0];

       for (i=0; i<15; i++) {
              suma += *p;
              p += stride[i];
       }
}
```

Indica el contingut inicial que ha de tenir el vector `stride` perquè els dos codis siguin equivalents.

```
int stride[15] = {
```

```
};
```

Pregunta 8. (1,20 punts)

La memòria virtual implementada en un sistema computador de 16 bits es caracteritza pels següents paràmetres:

- Pàgines d'1 KB de mida.
- Un màxim 10 pàgines carregades simultàniament a memòria física per aplicació.
- Reemplaçament de pàgines a memòria física seguint l'algorisme LRU.

Donat el següent codi (anàleg a la versió B de l'exercici 7, però amb una matriu de 32x32):

```
/* versió B */
int M[32][32];
int sum = 0;
int stride[528] = { ... };

main() {
int i;
int *p = &M[0][0];

       for (i=0; i<528; i++) {
              sum += *p;
              p += stride[i];
       }
}
```

Considera que el vector `stride` ja està inicialitzat correctament per tal que es recorri la matriu triangular inferior d'`M`, que les variables globals s'emmagatzemen a memòria a partir de l'adreça `0x0000`, i que el codi s'emmagatzema a partir de l'adreça `0xC000`. Es demana:

Raona quin serà el nombre total de fallades de pàgina que es produiran en executar aquest codi en el sistema computador esmentat.

Indica els VPN (en hexadecimal) de les pàgines que hi haurà carregades a memòria física quan s'acabi l'execució d'aquest programa (escriu-los en ordre de més petit a més gran).

COGNOMS:	**GRUP:**
NOM:	

Pregunta 9. (1,20 punts)

Un sistema disposa d'un processador de 20 bits d'adreces, i una memòria cache (MC) de 8 Kbytes amb la següent organització:

- Correspondència associativa per conjunts, de grau 2 (2 blocs per conjunt)
- Blocs de 256 bytes
- Reemplaçament LRU
- Escriptura retardada amb assignació.

Estant la cache inicialment buida, un programa produeix una seqüència de referències a memòria segons s'indica a la següent taula, on apareixen les adreces en hexadecimal i si són lectures o escriptures (L/E). Completa les columnes que falten indicant, per a cada referència: el número de conjunt de MC; si és encert (e) o fallada (f); i el nombre de bytes de Memòria Principal (MP) llegits i/o escrits.

L/E	adreça (hex)	núm. de conjunt	encert (e)/ fallada (f)	bytes de MP	
				llegits	escrits
L	01200				
E	082A0				
L	083F0				
E	01204				
L	04204				
E	083F4				
L	01208				
E	082A0				

Pregunta 10. (1,00 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.-	Si en una cache canviem la política d'escriptura *immediata amb assignació* a *retardada amb assignació*, sense cap més canvi, el nombre total de fallades no canvia.		
2.-	En memòria virtual paginada, sempre que reemplacem de la memòria física una pàgina cal escriure-la en disc.		
3.-	En un sistema amb memòria virtual, la mida total d'un programa i les seves dades no pot excedir la capacitat de la memòria física.		
4.-	Si un cert número enter de 8 bits es representa en Ca2 per 0x8B, el mateix número es representa en Ca1 per 0x8C		
5.-	Si el resultat d'un càlcul en format normalitzat de coma flotant causa underflow, es pot reduir l'error absolut de precisió si el podem expressar en format denormal.		
6.-	A l'inici de la rutina genèrica de servei d'excepcions de MIPS (RSE) aquesta sols ha de salvar a la pila aquells registres segurs que es modifiquin durant l'execució de la RSE.		
7.-	Un programa en mode usuari pot copiar un registre qualsevol de la CPU al coprocessador CP0 per mitjà de la instrucció `mtc0`.		
8.-	El tractament de les excepcions al MIPS es fa únicament en dues rutines de servei, una per a la fallada de TLB i una altra per a la resta d'excepcions.		
9.-	En el MIPS, el camp IM (Interrupt Mask) del registre Status usat en la gestió de les interrupcions serveix per indicar les peticions d'interrupció que no han de ser ateses quan s'acabi l'execució de la instrucció actual.		
10.-	La codificació *en excés* dels exponents dels nombres en coma flotant permet que la comparació de les seves magnituds es pugui fer amb un comparador de nombres naturals.		