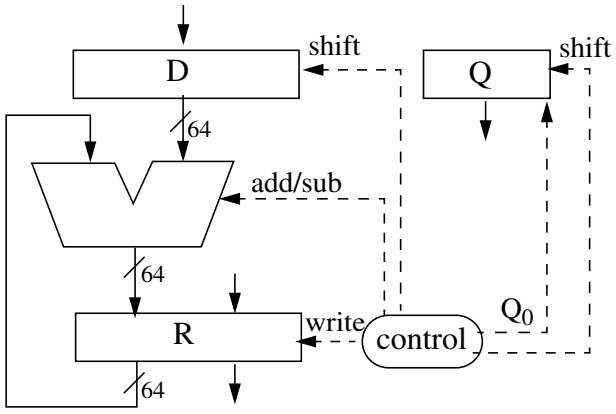
COGNOMS:	**GRUP:**
NOM:	

EXAMEN PARCIAL D'EC
6 de maig de 2019

L'examen consta de **8** 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 durada de l'examen és de 120 minuts. Les notes, la solució i el procediment de revisió es publicaran al Racó el dia **20** de maig.

Pregunta 1. (1 punt)

El següent esquema representa un circuit seqüencial que realitza la divisió dels nombres naturals de 32 bits X/Y. Aquest circuit calcula alhora el quocient QUO i el residu RES seguint el procediment de divisió amb restauració que has estudiat a classe. Completa l'algorisme en pseudocodi de la dreta (anàleg al llenguatge C) que en descriu el funcionament cicle a cicle.




```
R63:32 = [          ] ; R31:0 = [          ];
D63:32 = [          ] ; D31:0 = [          ];
Q      = [          ] ;
for (i=1; i<= [      ] ; i++)
{
   [                  ]
   [                  ]
   [                  ]
}
QUO    = [          ] ;
RES    = [          ] ;
```

Pregunta 2. (1 punt)

Suposem els valors inicials dels registres \$f2 = 0x41000001, i \$f4 = 0xBF40000F. Calcula el valor de \$f0 en hexadecimal després d'executar la instrucció `add.s $f0,$f2,$f4`.

\$f0 = 0x

Pregunta 3. (2 punts)

S'executa un programa de test en un ordinador que té 3 tipus d'instruccions (A,B,C), amb CPI diferents. La següent taula especifica el nombre d'instruccions de cada tipus que executa el programa i el seu CPI.

Tipus d'instrucció	Nombre d'instruccions	CPI
A	$8 \cdot 10^9$	7
B	$6 \cdot 10^9$	5
C	$4 \cdot 10^9$	4

a) Sabem que la freqüència de rellotge és de 1,5GHz i que la potència dissipada és P=100W. Calcula el temps d'execució en segons i l'energia consumida en Joules durant l'execució del programa de test.

t_{exe} = ____ s

E = ____ J

b) Es vol modernitzar l'equipament comprant un nou ordinador amb una CPU amb ISA diferent. Aquesta nova CPU funciona amb el mateix voltatge però te més transistors, així que la seva capacitància agregada (C) és un 50% més gran i el factor d'activitat (α) un 20% superior. A més a més, els seu ISA té 4 tipus d'instruccions (A,B,C,D). Un cop recompilat el programa, el nombre d'instruccions de cada tipus que executa i el seu CPI són les indicades en la següent taula:

Tipus d'instrucció	Nombre d'instruccions	CPI
A	$5 \cdot 10^9$	2
B	$3 \cdot 10^9$	4
C	$2 \cdot 10^9$	5
D	$2 \cdot 10^9$	1

Sabem que amb aquesta nova CPU obtenim un *speedup* de 2. A quina freqüència (en GHz) va la nova CPU? ¿Quina potència dissipada en Watts consumeix l'execució del programa (podem suposar que la potència estàtica és zero tant en la CPU original com en la nova)?

Freq = ____ GHz

P = ____ W

COGNOMS:	**GRUP:**
NOM:	

Pregunta 4. (2 punts)

Donada la següent funció `foo` en llenguatge C, correctament programada:

```
void foo(unsigned char k, char M[][100]) {
        int i;
        for (i=2; i<48; i+=4){
                M[50-i][i*2] = M[k][16];
        }
}
```

Completa el següent codi MIPS omplint les caselles en blanc perquè sigui equivalent a l'anterior codi en alt nivell, tenint en compte que els elements de la matriu M s'accedeixen utilitzant la tècnica **d'accés seqüencial** sempre que es pot, usant el registre `$t1` com a punter per a la matriu M. Aquest punter `$t1` s'inicialitza amb l'adreça de l'element `M[48][4]`. Al codi s'hi ha aplicat l'optimització de conversió d'un bucle `for` en un `do_while` i l'eliminació de la variable d'inducció.

```
        li    $t4, 100
        mult  $a0,$t4                 # k*100
        mflo  $t4
        addu  $t4, $a1, $t4           # @M[k][0]
        lb    $t4, [          ]($t4)  # $t4=M[k][16]
        addiu $t1, $a1, [          ]  # @ inicial del punter: $t1 = @M[48][4]
        addiu $t2, $a1, [          ]  # @ final del punter
        b     cond
bucle:  sb    $t4, [          ]($t1)
        addiu $t1, $t1, [          ]
cond:   bgeu  $t1, $t2, bucle
        jr    $ra
```

Pregunta 5. (1 punt)

Donada la següent sentència escrita en alt nivell en C:

```
if (((a<=b)&&(b!=0)) || (((b%8)^0x0005)>0))
        z=5;
else
        z=a-b;
```

Completa el següent fragment de codi MIPS, que tradueix l'anterior sentència, escrivint en cada calaix un mnemònic d'instrucció o macro, etiqueta, registre o immediat. Les variables `a`, `b` i `z` són de tipus `int` i estan inicialitzades i guardades als registres `$t0`, `$t1` i `$t2`, respectivament.

```
        [          ] $t0, $t1, [          ]
etq1:   [          ] $t1, $zero, [          ]
etq2:   andi       $t3, $t1, [          ]
etq3:   [          ] $t5, $t3, [          ]
etq4:   ble        $t5, $zero, [          ]
etq5:   li         $t2, 5
etq6:   b          [          ]
etq7:   subu       $t2, $t0, $t1
etq8:
```

Pregunta 6. (0,5 punts)

Donades les següents declaracions de funcions en C:

```
float g(int *w, int m);                 /* prototip de la funció g */
float f(int n) {
        float s, t;
        int v[100];

        for (s=0.0; s >= 0.0; n++) {
                t = g(v, n);
                s = t + g(v, n);
        }
        return s;
}
```

D'acord amb les regles de l'ABI estudiades, ¿quins elements de la funció f (paràmetres, variables locals i càlculs intermedis) s'han de guardar necessàriament en registres de tipus segur $s o $f? Especifica el registre segur concret que tries en aquells casos que en cal un. (Omple tantes entrades de la taula següent com et calguin):

Element de f (en C)	Registre

COGNOMS:	**GRUP:**
NOM:	

Pregunta 7. (1,5 punts)

Donades les següents declaracions de funcions en C:

```
void g(short *a, char *b, int c);
char f(int n) {
        char c;
        short v[20];

        g(v, &c, n);
        return c;
}
```

Tradueix a MIPS la funció f:

```
f:
```

Pregunta 8. (1 punt)

Un programa està compost de dos mòduls que es compilen i assemblen separadament per generar sengles fitxers objecte. Per a generar l'executable cal enllaçar-los després amb el muntador. El codi en C de les funcions `main()` i `func()` dels dos mòduls és el següent:

MÒDUL 1: `int main() { i=5; do { func(VA,i); i--} while(i!=0); }`
MÒDUL 2: `void func(int *W,int i) { VB[i]=W[i]+X; }`

Les variables `VA,X,VB` són globals de tipus `int` i estan definides en el mòdul 1. Hem traduït els dos fitxers a MIPS amb el següent resultat (hem afegit a l'esquerra els números de línia per facilitar les respostes posteriors):

MÒDUL 1

```
1         .data
2
3
4   VA:   .word -1, 0, -2, 0, -3
5   X:    .word 7
6   VB:   .word 0, 0, 0, 0, 0
7
8         .text
9         .globl main
10
11  main: addiu  $sp, $sp, -8
12        sw     $s0, 0($sp)
13        sw     $ra, 4($sp)
14        li     $s0, 5
15  do:
16        la     $a0, VA
17        move   $a1, $s0
18        jal    func
19        addiu  $s0, $s0, -1
20        bne    $s0, $zero, do
21        lw     $s0, 0($sp)
22        lw     $ra, 4($sp)
23        addiu  $sp, $sp, 8
24        jr     $ra
```

MÒDUL 2

```
1         .data
2
3
4         .text
5
6   func: sll    $t0, $a1, 2
7         addu   $t1, $a0, $t0
8         lw     $t2, 0($t1)
9
10        la     $t3, X
11        lw     $t4, 0($t3)
12        addu   $t4, $t4, $t2
13
14        la     $t5, VB
15        addu   $t6, $t5, $t0
16        sw     $t4, 0($t6)
17        jr     $ra
```

a) Quan hem intentat enllaçar els dos fitxers objecte generats per l'assemblador, l'enllaçador ha detectat diversos errors del tipus "referència no-resolta". ¿Com caldrà corregir o completar el codi MIPS perquè no torni a fallar? (fes servir tantes files de la taula com necessitis)

mòdul	núm. línia	línia de codi

b) Contesta les següents preguntes suposant que s'han tornat a assemblar els dos fitxers després de corregir els errors i que totes les instruccions conserven la numeració de línies original:

Pregunta	MÒDUL 1	MÒDUL 2
Quines etiquetes conté la Taula de Símbols Globals en cada fitxer objecte?		
Quines instruccions en cada fitxer objecte (sols el número de línia) requereixen **adreces absolutes** (de dades o de codi)?		
Quines instruccions en cada fitxer objecte (sols el número de línia) contenen **referències no-resoltes** (a dades o a codi definits en l'altre mòdul)?		