COGNOMS:	**GRUP:**
NOM:	

EXAMEN PARCIAL D'EC
8 de novembre de 2018

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 novembre.

Pregunta 1. (1,0 punts)

Donada la següent sentència escrita en alt nivell en C:

```
if (((a & 0xFFFF) != 0) && ((a ^ 0xAAAA) > 0))
    a = 5;
else
    a = 1;
```

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. La variable `a` és de tipus `int` i està inicialitzada i guardada al registre `$t0`.

```
        [        ] $t1, $t0, 0xFFFF
        [        ] $t1, $zero, [    ]
etq1:   [        ] $t1, $t0, 0xAAAA
etq2:   [        ] $t1, $zero, [    ]

etq3:   li      $t0, 5

etq4:   b       et6

etq5:   li      $t0, 1

etq6:
```

Pregunta 2. (1,0 punts)

Suposem que els registres MIPS `$t1` i `$t2` valen `$t1=0x12341234` i `$t2=0xFFFFFFFF`. Indica el contingut final en hexadecimal de `$hi` i `$lo` després d'executar la instrucció `div $t1,$t2` i després de la instrucció `divu $t1,$t2`.

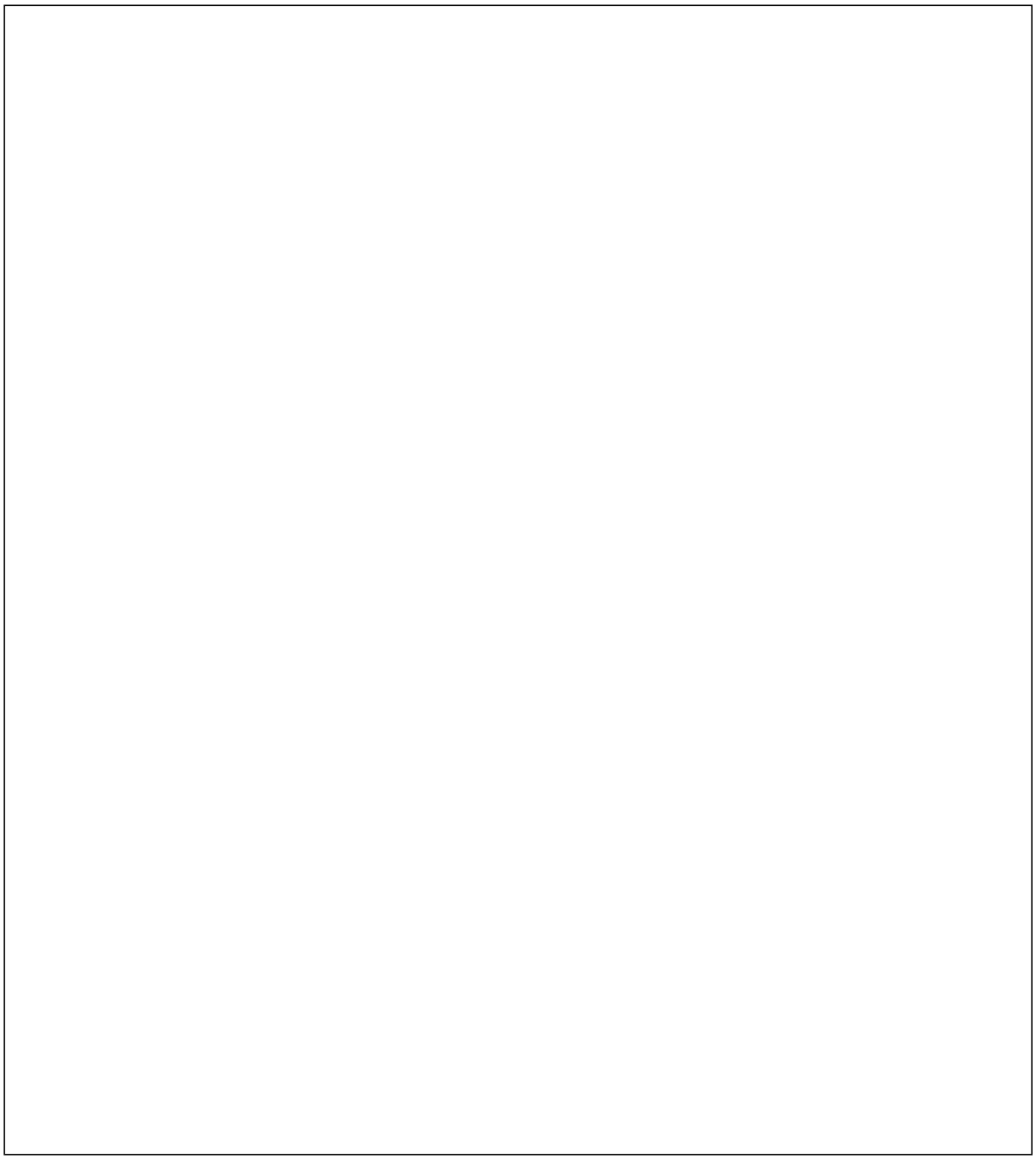
`div $t1,$t2`	`$hi =`	`0x`	`$lo =`	`0x`
`divu $t1,$t2`	`$hi =`	`0x`	`$lo =`	`0x`

Pregunta 3. (1,5 punts)

Tradueix la funció `f2` de llenguatge C a una subrutina MIPS:

```
void f1(int v[], int x);
int f2(int a, int b)
{
    int tmp[2];

    f1(tmp, a);
    return tmp[0] + tmp[1] + b;
}
```



COGNOMS:	GRUP:
NOM:	

Pregunta 4. (1,5 punts)

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

```
char a[3] = "AB";
long long int b = -3;
char *c = &a[1];
char d = 'C';
int e[2] = {1025, 17};
```

a) Tradueix-la al llenguatge assemblador del MIPS.

b) 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	
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) Donat el següent codi en assemblador MIPS, indica quin és el valor final en hexadecimal del registre `$t0`:

```
la      $t0, c
lw      $t0, 0($t0)
lb      $t0, 0($t0)
sra     $t0, $t0, 4
```

$t0 = 0x

Pregunta 5. (1,5 punts)

Donada la següent declaració en C:

```
void func(int mat[][40], int i) {
        mat[i][20-i] = mat[20-i][i];
}
```

Completa els requadres del següent fragment de codi per tal que sigui la traducció optimitzada de la funció func (pista: determina les dues adreces de memòria en funció de `i`, i observa si contenen algun terme en comú):

```
func:   li      $t2,
        mult    $t2, $a1
        mflo    $t2
        subu    $t1, $a0, $t2
        addu    $t0, $a0, $t2
        lw      $t3,            ($t1)
        sw      $t3,            ($t0)
        jr      $ra
```

Pregunta 6. (1,0 punts)

Donades les següents declaracions en C:

```
int vec[100];
void f(short *p) {
        *(p+5) = *p + 3;
}

int *g(int i) {
        return &vec[i];
}
```

a) Tradueix a MIPS la funció f

b) Tradueix a MIPS la funció g

COGNOMS:	**GRUP:**
NOM:	

Pregunta 7. (1,5 punts)

Considera la següent funció f, que rep com paràmetres en $a0 i $a1 dos vectors de 1000 enters, i que retorna el seu producte escalar.

```
f:      addiu   $t0, $zero, 0
        addiu   $v0, $zero, 0
do:
        lw      $t1, 0($a0)
        lw      $t2, 0($a1)
        mult    $t1, $t2
        mflo    $t3
        addu    $v0, $v0, $t3
        addiu   $a0, $a0, 4
        addiu   $a1, $a1, 4
        addiu   $t0, $t0, 1
        slti    $t4, $t0, 1000
        bne     $t4, $zero, do      # salta si $t0<1000

        jr      $ra
```

Suposem que s'executa en un processador amb un rellotge de 2GHz i amb els següents CPI segons el tipus d'instrucció:

TIPUS	multiplicació	salt (salta)	salt (no salta)	load/store	altres
CPI	9	5	1	5	1

Els càlculs que es demanen a continuació sobre l'execució de la funció `f` s'han d'expressar amb nombres decimals amb un màxim de 2 dígits fraccionaris de precisió (es poden obtenir fàcilment sense calculadora).

a) Calcula el temps d'execució de `f` en microsegons ($1\mu s = 10^{-6}$ s)

temps = _____ µs

b) Calcula el CPI promig

CPI = _____

c) Calcula el guany de rendiment (speedup) que obtindríem si optimitzem el disseny de la CPU de manera que les multiplicacions tardin 4 cicles en lloc de 9.

guany = _____

Pregunta 8. (1,0 punts)

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 dels dos mòduls és el següent:

MODUL 1: `int main() { f(V[X]); }`

MODUL 2: `void f(int i) { Y=i; }`

Les variables `V`, `X`, `Y` són globals. 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        .globl V
3   V:   .word 1, 2, 3, 4, 5
4   Y:   .word 0

5        .text
6        .globl main
7   main:addiu  $sp, $sp, -4
8        sw     $ra, 0($sp)

9        la     $t0, X
10       lw     $t1, 0($t0) # $t1 <- X
11       la     $t2, V
12       sll    $t3, $t1, 2
13       addu   $t4, $t2, $t3
14       lw     $a0, 0($t4) # $a0 <- V[X]
15       jal    f

16       lw     $ra, 0($sp)
17       addiu  $sp, $sp, 4
18       jr     $ra
```

MÒDUL 2

```
1        .data
2        .globl  X
3   X:   .word   3

4        .text
5        .globl  f
6   f:   la     $t0, Y
7        sw     $a0, 0($t0) # Y <- $a0
8        jr     $ra
```

a) Quan hem intentat enllaçar els dos fitxers objecte generats per l'assemblador, l'enllaçador ha detectat un error. Com caldrà corregir el codi MIPS perquè no torni a fallar?

mòdul:
codi corregit:

b) Contesta les següents 3 preguntes suposant que s'ha corregit l'error anterior i que conservem la numeració de línies original:

Pregunta	MÒDUL 1	MÒDUL 2
Quines etiquetes conté la Taula de Símbols Globals de cada fitxer objecte?		
Quines línies de codi en cada fitxer objecte (sols el número) contenen referències **no-resoltes** (referències creuades)?		
Quines línies de codi en cada fitxer objecte (sols el número) contenen **adreces absolutes** (i necessiten ser reubicades)?		