

COGNOMS:

GRUP:

NOM:

EXAMEN FINAL D'EC

10 de gener de 2019

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 durada de l'examen és de 180 minuts. Les notes, la solució i el procediment de revisió es publicaran al Racó abans del dia 17 de gener.

Pregunta 1. (1,6 punts)

Donades les següents declaracions de variables globals en ensamblador del MIPS, que s'ubiquen en memòria a partir de l'adreça 0x10010000:

```
.data
a: .asciiz "abcd"           # codi ASCII 'a' = 0x61
b: .half 10, -7
c: .word b
d: .byte 0x03
   .align 1
e: .space 4
f: .word 256
```

- a) (0,4 pts) Omple la següent taula amb el contingut de la memòria, indicant el valor de cada byte EN HEXADECIMAL, i deixant EN BLANC les posicions no ocupades per cap dada.

@Memòria	Dada	@Memòria	Dada	@Memòria	Dada	@Memòria	Dada
0x10010000		0x10010008		0x10010010		0x10010018	
0x10010001		0x10010009		0x10010011		0x10010019	
0x10010002		0x1001000A		0x10010012		0x1001001A	
0x10010003		0x1001000B		0x10010013		0x1001001B	
0x10010004		0x1001000C		0x10010014		0x1001001C	
0x10010005		0x1001000D		0x10010015		0x1001001D	
0x10010006		0x1001000E		0x10010016		0x1001001E	
0x10010007		0x1001000F		0x10010017		0x1001001F	

- b) (0,4 pts) Quin és el valor de \$t0 en hexadecimal després d'executar el següent codi?

```
la    $t0, c
lw    $t0, 0($t0)
lh    $t0, 2($t0)
```

\$t0 =

0x

- c) (0,4 pts) Quin és el valor final de \$t0 i de \$t1 en hexadecimal després d'executar el següent codi?

```
li    $t0, 2563
li    $t1, 10
div    $t0, $t1
mflo   $t0
mfhi   $t1
```

\$t0 =

0x

\$t1 =

0x

d) (0,4 pts) Quin és el valor final de $\$t0$ en hexadecimal després d'executar el següent codi?

```
li      $t0, -3
sra     $t1, $t0, 31
xor     $t0, $t0, $t1
subu    $t0, $t0, $t1
sll     $t1, $t1, 31
or      $t0, $t0, $t1
```

$\$t0 =$

Pregunta 2. (1,3 punts)

Considerem un computador amb un processador MIPS funcionant a una freqüència de 0,5Ghz, i que dissipa una potència de 10 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 $t_{block} = 99$ cicles. Els CPI dels diversos tipus d'instruccions (en absència de fallades) són:

	Salts	Loads	Resta d'instruccions
CPI	3	5	2

A través de simulacions amb un programa de test hem mesurat una taxa de fallades de la cache de dades del 4,4% (és a dir, $m = 0,044$). Totes les referències a memòria són lectures. El nombre d'instruccions executades és:

	Salts	Loads	Resta d'instruccions
núm. instr.	$1 \cdot 10^9$	$3 \cdot 10^9$	$6 \cdot 10^9$

a) (0,4 pts) Calcula el CPI_{ideal} del programa (CPI promig amb cache ideal sense fallades)

$CPI_{ideal} =$

b) (0,4 pts) Calcula, en segons, el temps d'execució (incloent-hi fallades de cache)

$t_{exe} =$ s

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

$E =$ J

d) (0,4 pts) Calcula, en cicles, el temps d'accés mitjà a memòria dels loads per a aquest programa

$t_{am} =$ cicles

COGNOMS:

GRUP:

NOM:

Pregunta 3. (1,4 punts)

Donades les següents declaracions en C:

```
int *pglob; /* variable global */
int f2(int x, char *y, char *z); /* prototipus de f2 */
char f1(int a, char b[][5], int c, int *d) {
    char v[10];
    if ((c < 0) || (c >= a))
        c = 0;
    *pglob = f2(*d, &b[3][0], v);
    return v[c];
}
```

A continuació es mostra una traducció de la funció f1 a llenguatge MIPS que està incompleta. Llegiu-la amb atenció, i completeu les caixes per tal que la traducció sigui correcta.

```
f1:      addiu    $sp, $sp, -20
        sw       $s0, 12($sp)
        sw       $ra, 16($sp)
        [ ]      $a2, $zero, [ ]          # c<0?
        [ ]      $a2, $a0, [ ]           # c>=a?
then:
        move     $a2, $zero
endif:
        move     $s0, [ ]                # copiar en registre segur
        # Passar paràmetres: f2(*d, &b[3][0], v)
        [ ]
        jal f2
        # Emmagatzemar resultat: *pglob = f2(...)
        [ ]
        # Sentència final: return v[c];
        [ ]
        lw       $s0, 12($sp)
        lw       $ra, 16($sp)
        addiu    $sp, $sp, 20
        jr       $ra
```

Pregunta 4. (1,2 punts)

Considera la següent declaració MIPS de variables globals:

```
a:      .word 0xCC800000
b:      .word 0x4C800000
c:      .float 1.0
```

Suposant que s'executa el següent codi:

```
la      $t0, a
lwc1    $f0, 0($t0)
la      $t0, b
lwc1    $f2, 0($t0)
la      $t0, c
lwc1    $f4, 0($t0)
```

Es demana que contesteu quin serà el valor final a \$f6 en hexadecimal després de l'execució dels següent codis:

a) (0,6 pts)

```
add.s   $f6, $f0, $f2
add.s   $f6, $f6, $f4
```

\$f6 =

b) (0,6 pts)

```
add.s   $f6, $f2, $f4
add.s   $f6, $f0, $f6
```

\$f6 =

COGNOMS:

GRUP:

NOM:

Pregunta 5. (0,8 punts)

Considerant la declaració de la matriu global A

```
int A[N][N];
```

el següent codi en llenguatge C copia la triangular superior cap a la inferior d'aquesta matriu quadrada:

```
int i,j;

for (i=0; i<N; i++)
    for (j=i+1; j<N; j++)
        A[j][i] = A[i][j];
```

A continuació tenim un codi incomplet també en llenguatge C que és una optimització del codi anterior:

```
int *pdi,*pei,*pdj,*pej;

pdi = &A[0][1];
pei = &A[1][0];
do {
    pdj = pdi;
    pej = pei;
    do {
        *pej = *pdj;

        pdj +=  ;

        pej +=  ;

    } while (pej <= &A[N-1][N-1]);

    pdi +=  ;

    pei +=  ;

} while (pdi <= &A[N-1][N-1]);
```

S'hi ha aplicat l'optimització de convertir un bucle `while` en un `do_while`, d'eliminar variables d'inducció i de fer accés seqüencial tant per l'accés a `A[i][j]` com per a `A[j][i]`.

Es demana que completeu el codi anterior.

Pregunta 6. (0,5 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.-	En un programa escrit en assembler MIPS, que consta de 2 mòduls A i B que es compilen per separat, i on el mòdul A invoca una funció <code>func</code> que està declarada al mòdul B, l'assemblatge del mòdul A fallarà sense generar cap fitxer objecte.		
2.-	Si l'accés a dades d'una instrucció produeix un encert al TLB, però el bit V val 0, llavors la instrucció causarà una excepció de fallada de pàgina.		
3.-	Una mateixa instrucció pot causar durant la seva execució 2 fallades de pàgina.		
4.-	La instrucció <code>tlbwr</code> escriu una entrada de la taula de pàgines a l'entrada del TLB que resulta d'aplicar un algorisme de reemplaçament RANDOM.		
5.-	Si el bit EXL (Exception Level) del registre Status val 1, el processador està en mode sistema i totes les excepcions resten inhibides.		

COGNOMS:

GRUP:

NOM:

Pregunta 7. (1,2 punts)

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

- Pàgines de 4 KB 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 8 entrades amb reemplaçament LRU.

Donat el següent codi en C:

```
int V[8192];

main() {
    int i;
    int sum = 0;

    for (i=0; i < 8192; i++) {
        sum += V[i] + V[8191 - i];
    }
}
```

Considera que les variables locals *i* i *sum* s'emmagatzemen en registres, que el vector global *V* s'emmagatzema a memòria a partir de l'adreça 0x00000000, i que el codi s'emmagatzema a partir de l'adreça 0x0000C000, i ocupa menys d'una pàgina. El TLB i la memòria física estan inicialment buits. Es demana:

a) (0,3 pts) Quantes pàgines ocupa el vector *V*?

nombre de pàgines =

b) (0,3 pts) Quantes fallades de TLB (codi i dades) es produiran en tota l'execució del programa?

fallades de TLB =

c) (0,3 pts) Quantes fallades de pàgina (codi i dades) es produiran en tota l'execució del programa?

fallades de pàgina =

d) (0,3 pts) 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 d'aquest programa.

VPNs =

Pregunta 8. (1,0 punts)

- a) (0,4 pts) Escriu 4 formes equivalents d'expandir la macro `li $t0,1` usant únicament 1 instrucció.

- b) (0,3 pts) Escriu en decimal i en notació científica normalitzada de base 2 el menor número real positiu tal que en l'estàndard IEEE-754 de simple precisió es codifiqui com un denormal.

--

- c) (0,3 pts) Donat el següent codi en C que usa la variable `a` de tipus `int`:

```
a = ((a & 1) && 1) || 1 | 1;
```

i considerant que la variable `a` està emmagatzemada al registre `$t0`, escriu un codi MIPS equivalent amb 2 línies com a màxim.

--

Pregunta 9. (1,0 punts)

Un sistema disposa d'un processador MIPS (32 bits d'adreces i mida de paraula de 4 bytes), i una memòria cache (MC) de 64 Kbytes amb la següent organització:

- Correspondència directa
- Blocs de 256 bytes
- Escriptura immediata sense assignació

Estant la cache inicialment buida, un programa fa 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: l'índex 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)	índex MC	encert (e)/ fallada (f)	bytes de MP	
				llegits	escrits
E	00010000				
L	00010008				
E	00010016				
L	00F40316				
E	03100004				
L	00010024				
L	03200308				
L	00F403A8				