

EXAMEN PARCIAL D'EC

29 de març de 2020

INSTRUCCIONS IMPORTANTS:

- La durada de l'examen és de 120 minuts.
- L'examen consta de 8 preguntes, que s'han de contestar al fitxer de respostes (el fitxer de text respostes.txt que acompanya a aquest enunciat).
- A la capçalera del fitxer de respostes posa el teu nom i cognoms. No canviïs el nom del fitxer de respostes.
- El fitxer de respostes s'ha de pujar al Racó, al mateix lloc des d'on has descarregat aquest enunciat (dins l'apartat de Pràctiques i l'apartat habilitat per a l'examen parcial del grup 40).
- Pots pujar el fitxer múltiples cops, no esperis a l'últim minut.

Pregunta 1 (0,5 punts)

Donada la següent subrutina en ensamblador MIPS:

```
acces_aleatori:
    li    $t0, 400
    mult  $t0, $a1
    mflo  $t0
    addu  $t0, $t0, $a0
    lw    $v0, 480($t0)
    jr    $ra
```

Sabem que és la traducció de la següent funció en C (de la qual desconeixem els valors dels requadres). Completa els requadres amb les expressions vàlides en C per tal que la traducció sigui correcta:

```
int acces_aleatori (int M[][100], int i)
{
    return M[ ] [ ];
}
```

Pregunta 2 (1 punt)

Donada la següent funció `foo` en alt nivell correcte (no cal comprovar si se surt de rang), on `M` i `V` són declarades com a variables globals:

```
int M[100][100]; /* suposarem que està inicialitzada */
int V[100];
void foo() {
    int i, aux; /* ocupen els registres $t1 i $t3 respectivament */
    for (i=0; i<97; i++) {
        aux = M[i][i+3];
        M[i+1][i+3] = V[aux];
    }
}
```

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 s'accedeixen utilitzant la tècnica **d'accés seqüencial**, usant el registre `$t0` com a punter per accedir els elements `M[i][i+3]`.

```
la    $t0, M + 
li    $t1, 0
li    $t2, 97
la    $t7, V
bucle: bge  $t1, $t2, fibuc
lw    $t3, ( $t0)
sll   $t3, $t3, 2
addu  $t3, $t3, $t7
lw    $t4, ( $t3)
sw    $t4, ( $t0)
addiu $t0, $t0, 
addiu $t1, $t1, 1
b bucle
fibuc:
```

Ens han demanat que dissenyem un processador compatible amb l'ISA del MIPS per a una aplicació específica. Per a obtenir un millor rendiment, aquesta aplicació necessita un format per a coma flotant en simple precisió una mica lleuger a l'estàndard IEEE-754. El nou format serà similar excepte que té 6 bits per l'exponent, codificat en excés 31, i 25 bits per la mantissa.

```
li      $t1, 0x48040002
mtc1    $t1, $f4
li      $t2, 0xC0400009
mtc1    $t2, $f6
add.s   $f8, $f4, $f6
```

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

	mantissa(binari)	exponent (decimal)
\$f_4\$:	<div style="display: flex; align-items: center;"> <div style="width: 20px; height: 20px; margin-right: 5px;"></div> <div style="flex-grow: 1; border-top: 1px solid black; border-bottom: 1px solid black; position: relative;"> , </div> </div>	
\$f_6\$:	<div style="display: flex; align-items: center;"> <div style="width: 20px; height: 20px; margin-right: 5px;"></div> <div style="flex-grow: 1; border-top: 1px solid black; border-bottom: 1px solid black; position: relative;"> , </div> </div>	

																						G	R	S		
op	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

																				G	R	S	exponent (decimal)

[illegible]

```
$f8 = 0x
```

Donades les següents declaracions de variables locals

i el següent codi en alt nivell

Sabent que les variables `a`, `b`, `c`, `d` estan guardades en els registres `$t0`, `$t1`, `$t2`, `$t3` respectivament, completa el següent fragment de codi MIPS omplint les caselles en blanc perquè sigui equivalent a l'anterior codi en alt nivell (*nota: posa atenció al tipus de les variables!*):

```

et1:       $t0, $t1, 
et2:       $t0, $t2, 
et3:       $t3, $zero, 
et4:      li      $t3, -1
et5:      b       et7
et6:      move    $t3, $zero
et7:

```

Pregunta 5 (2 punts)

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

```
short a[] = {513, 17, -5};  
long long b = 1030;  
short c = 0;  
char d[] = "COVID19"; /* s'usen tants bytes com calgui per l'string*/  
short *e = &c;
```

a) Tradueix-la al llenguatge ensamblador del MIPS

b) Completa la següent taula amb el contingut de memòria en hexadecimal. Tingues en compte que el codi ASCII de la '0' és el 0x30, que el codi ASCII de la 'A' és el 0x41 i que el codi ASCII de la 'V' és el 0x56. Les variables s'emmagatzemen a partir de l'adreça 0x10010000. Les posicions de memòria sense inicialitzar es deixen en blanc.

@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	

c) Donat el següent codi en ensamblador MIPS, indica quin és el valor final en hexadecimal del registre \$t1:

```
la $t0, d  
lbu $t0, 5($t0)  
addiu $t0, $t0, -0x32  
sltu $t1, $zero, $t0
```

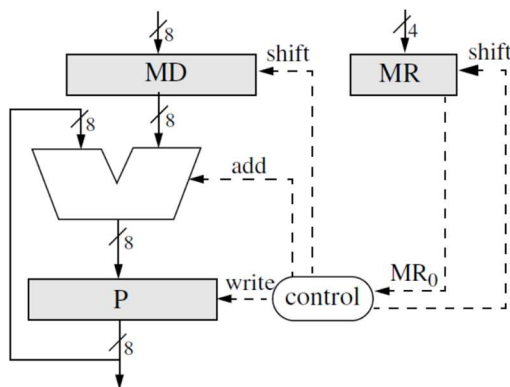
\$t1 = 0x

d) Tradueix a llenguatge ensamblador del MIPS la següent sentència en C:

```
*e = a[0] - a[2]
```

Pregunta 6 (1,5 punts)

Segui el següent diagrama del multiplicador seqüencial de nombres naturals de 4 bits anàleg al que s'ha estudiat durant el curs, el qual calcula el producte amb 8 bits:



- a) Suposem que amb aquest circuit multipliquem els números binaris de 4 bits 1011 (multiplicand) i 1010 (multiplicador). Completa la següent taula, que mostra els valors en binari dels registres P, MD, i MR després de la inicialització i després de cada iteració, afegint tantes iteracions com facin falta:

Iter.	P (producte)								MD (Multiplicand)								MR (Multiplicador)			
ini	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0
1																				
2																				
3																				
4																				

- b) Suposem que els registres MIPS \$t1 i \$t2 valen \$t1=0x00000008 i \$t2=0xFFFFFFFF. Indica el contingut final en hexadecimal de \$hi i \$lo després d'executar multu \$t1,\$t2 i després de mult \$t1,\$t2.

multu: \$hi= \$lo=

mult: \$hi= \$lo=

Pregunta 7 (1 punt)

Donades les següents declaracions de funcions en C:

```
float func1(int *w, int m); /* prototip de la funció func1 */
float func2(int n) {
    float s, t;
    int v[100];
    for (s=0.0; s >= 0.0; n++) {
        t = func1(v, n);
        s = t + func1(v, n);
    }
    return s;
}
```

- a) D'acord amb les regles de l'ABI estudiades, ¿quins elements de la funció func2 (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 func2 (en C)	Registre

- b) De quina mida en bytes seria el bloc d'activació de un cop traduïda a MIPS la funció func2?

Mida = bytes

Pregunta 8 (1,25 punts)

Un processador disposa de 3 tipus d'instruccions (A,B,C), amb diferents CPI. A fi de caracteritzar el seu rendiment i consum hi executem un programa de test. La següent taula especifica el nombre d'instruccions executades de cada tipus, així com el seu CPI.

Tipus d'instrucció	CPI	Nombre d'instruccions
A	1	$8 \cdot 10^9$
B	4	$3 \cdot 10^9$
C	5	$2 \cdot 10^9$

- a) Suposant que la potència dissipada és $P=60\text{W}$ i que la freqüència de rellotge és de $1,2\text{GHz}$, calcula el temps d'execució en segons i l'energia consumida en Joules durant l'execució del programa de test.

$$t_{\text{exe}} = \boxed{} \text{ s}$$

$$E = \boxed{} \text{ J}$$

- b) Hem redissenyat el processador, i ara la freqüència és de $1,8 \text{ GHz}$, però les instruccions de tipus B tenen un $\text{CPI}=6$. Suposant que no ha canviat cap més paràmetre arquitectònic o electrònic, calcula el temps d'execució en segons, el guany de rendiment (speedup) i la potència dissipada en Watts durant l'execució del programa de test.

$$t_{\text{exe}} = \boxed{} \text{ s}$$

$$\text{speedup} = \boxed{}$$

$$P = \boxed{} \text{ W}$$