Cognoms: …… Nom: ……
DNI:……

EXAMEN PARCIAL D'EC
4 de novembre de 2021

- L'examen consta de 5 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 1:30 hores (90 minuts)
- Les notes, la solució i el procediment de revisió es publicaran al Racó el dia 12 de novembre.

Pregunta 1 *(2,50 punts)*

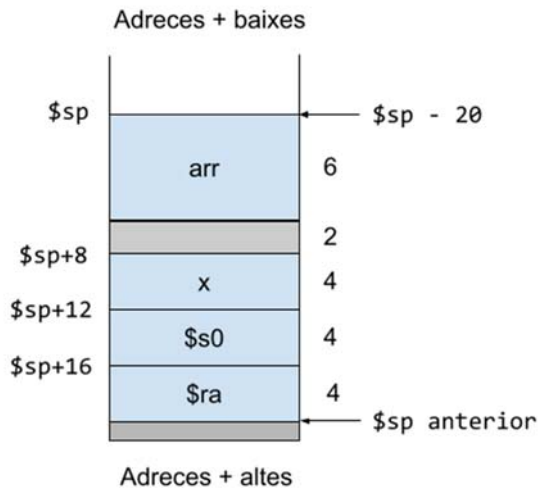
Donades les següents declaracions de funcions en C:

```
int f(short *a, int *b, char c);

int g(int j, char *k) {
    short arr[3];
    int x, y;

    y = j + f(&arr[1], &x, *k);
    return y * 2;
}
```

a) Dibuixa el bloc d'activació (stack frame) de `g`, indicant a quina posició apunta el registre `$sp` un cop reservat l'espai necessari a la pila, així com el nom de cada registre o variable que es guardi a la pila i la seva posició respecte a `$sp` (`$sp` + n bytes).



b) Tradueix el codi de la subrutina g.

```
g:
    addiu   $sp, $sp, -20
    sw      $s0, 12($sp)
    sw      $ra, 16($sp)
    move    $s0, $a0

    addiu   $a0, $sp, 2
    lb      $a2, 0($a1)
    addiu   $a1, $sp, 8
    jal     f
    addu    $v0, $v0, $s0
    addu    $v0, $v0, $v0

    lw      $ra, 16($sp)
    lw      $s0, 12($sp)
    addiu   $sp, $sp, 20
    jr      $ra
```

Cognoms: …… Nom: ……
DNI:……

Pregunta 2 *(1. 75 punts)*

Considera la següent subrutina programada en assemblador MIPS:

```
func:   ble   $a2, $a1, et1
        bgt   $a0, $a1, et2
et1:    blt   $a2, $zero, et4
et2:    move  $v0, $a0
et3:    b     et5
et4:    move  $v0, $a2
et5:    jr    $ra
```

Completa el següent codi escrit en C omplint les caselles en blanc perquè sigui equivalent a l'anterior codi en assemblador:

```
int func(int x, int y, int z) {
    int res;
    if ( (( z > y )&&( x > y )) || ( z >= 0 ) ) {
        res = x ;
    } else {
        res = z ;
    }
    return res;
    }
```

Pregunta 3 *(1.75 punts)*

Donat el següent codi en C que es tradueix a MIPS just a sota es demana que omplis les caselles buides del codi MIPS en funció de la constant N.

Codi C

```
#define N 1000
int m[N][N];

void main(){
  int i, suma=0;

  for (i=N-1; i>0; i-=2)
        suma += m[i][N-i];
}
```

Codi MIPS

```
main:   move  $t1, $zero
        li    $t0, N - 1
        la    $t2, m + 4 * N * (N-1) + 4
for:    lw    $t3, 0($t2)
        addu  $t1, $t1, $t3
        addiu $t0, $t0, -2
        addiu $t2, $t2, 8 * (1-N)
        bgt   $t0, $zero, for
        jr    $ra
```

Pregunta 4 *(2 punts)*

Donada la següent declaració de variables globals, que s'ubica a memòria a partir de l'adreça 0x10010000:

```
.data
a1:   .byte   '5'            # el codi ascii de '0' és 48
      .align  2
a2:   .space  3
a3:   .asciiz "2026"
a4:   .half   1, 0x37, -5
a5:   .word   a3
a6:   .half   0x7fff
```

a) Omple la següent taula amb el contingut de memòria **en hexadecimal**. Les posicions de memòria sense inicialitzar es deixen en blanc.

@Memòria	Dada	@Memòria	Dada	@Memòria	Dada	@Memòria	Dada
0x10010000	**0x35**	0x10010008	**0x30**	0x10010010	**0xFB**	0x10010018	**0xFF**
0x10010001		0x10010009	**0x32**	0x10010011	**0xFF**	0x10010019	**0x07**
0x10010002		0x1001000A	**0x36**	0x10010012		0x1001001A	
0x10010003		0x1001000B	**0x00**	0x10010013		0x1001001B	
0x10010004	**0x00**	0x1001000C	**0x01**	0x10010014	**0x07**	0x1001001C	
0x10010005	**0x00**	0x1001000D	**0x00**	0x10010015	**0x00**	0x1001001D	
0x10010006	**0x00**	0x1001000E	**0x37**	0x10010016	**0x01**	0x1001001E	
0x10010007	**0x32**	0x1001000F	**0x00**	0x10010017	**0x10**	0x1001001F	

b) Donat el següent codi que fa referència a l'anterior declaració:

```
main:
   la    $t0, a5
   lw    $t0,0($t0)
   lb    $t1,3($t0)
   la    $t2, a4
   lh    $t2, 4($t2)
   addu  $t1, $t1, $t2
   sb    $t1,3($t0)
   move  $t3, $zero
   li    $t4, 0x0a
   la    $t0, a3
   li    $t1, 3
loop:
   mult  $t3, $t4
   mflo  $t3
   lb    $t5, 0($t0)
   andi  $t5, $t5, 0x0f
   addu  $t3, $t3, $t5
   addiu $t1, $t1, -1
   addiu $t0, $t0, 1
   ble   $zero, $t1, loop
   jr    $ra
```

Omple la següent taula amb el valor en decimal dels registres $t1 i $t3 just ABANS d'executar la instrucció en negreta (cal usar una fila de la taula per cada iteració que es faci) i els valors dels mateixos registres en sortir del bucle:

	$t1	**$t3**
1a iter.:	**3**	**0**
2a iter.:	**2**	**2**
...	**1**	**20**
	0	**202**
en sortir:	**-1**	**2021**

Cognoms: …… Nom: ……
DNI:……

Pregunta 5 *(2 punts)*

Hem executat un programa que estem analitzant en un nou processador MIPS que funciona a una freqüència de 2 GHz. El programa té la següent distribució d'instruccions segons el seu CPI:

Tipus	CPI	% Instruccions
Accés a memòria (load/store)	8	20 %
Aritmètiques (add/sub/…)	2	50 %
Branches	4	30 %

Sabem que el nombre total d'instruccions del programa és de 10^9, i que la potència que dissipa el processador és de 100W.

a) Quin temps d'execució té el nostre programa en aquesta CPU (en segons), i quina quantitat d'energia necessitarà la seva execució (en Joules)?

Temps (s)	**1,9 s**
Energia (J)	**190 J**

Els nostres enginyers diuen que abans de llençar el nou processador MIPS al mercat hi ha temps per reduir el CPI d'un dels tipus d'instrucció a la meitat ($CPI_{nou} = CPI_{vell} / 2$).

b) Quin CPI hauríem de reduir per obtenir el màxim speedup en l'execució del nostre programa? Quin speedup obtindríem (pots deixar-ho en format de fracció)?

Instrucció a millorar	**Memòria**
Speedup obtingut	**~1,27 (38/30 o 19/15)**

Com a resultat de millorar el CPI, per mantenir la mateixa freqüència els enginyers han estimat que caldrà augmentar el voltatge del processador un 10%.

c) Suposant que la potència estàtica que dissipa el processador és zero (abans i després de la millora), quina serà la nova potència dissipada pel processador? Quanta energia consumirem ara en executar el nostre programa?

Potència (W)	**121 W**
Energia (J)	**181,5 J**