COGNOMS, NOM:

EXAMEN PARCIAL D'EC

4 de novembre de 2020

L'examen consta de 5 preguntes, que s'han de contestar als mateixos fulls de l'enunciat. No oblideu posar el nom i cognoms a tots els fulls. La durada de l'examen és de 90 minuts. Les notes, la solució i el procediment de revisió es publicaran al Racó.

Problema 1. (2,5 punts)

Donades les següents declaracions en llenguatge C, traduïu a assemblador MIPS la funció `f1` com una subrutina.

```
int f2(char a, long long *b, char c, int d);

int f1(long long w[], int i, char *p, char c) {
      return f2(*(p+i), &w[i], c, i) + 4*i;
}
```

```
f1:
      addiu  $sp, $sp, -8
      sw     $s0, 0($sp)
      sw     $ra, 4($sp)
      move   $s0, $a1          # guardem la i

      sll    $t0, $s0, 3
      addu   $a1, $a0, $t0     # 2n param: l'adreça és w + i*8
      addu   $a0, $a2, $s0     #
      lw     $a0, 0($a0)       # 1r param: contingut de l'@ p+i*1
      move   $a2, $a3          # 3r param: c
      move   $a3, $s0          # 4t param: i
      jal    f2
      sll    $t0, $s0, 2       # 4 * i
      addu   $v0, $v0, $t0     # retorna f2() + 4*i

      lw     $s0, 0($sp)
      lw     $ra, 4($sp)
      addiu  $sp, $sp, 8
      jr     $ra
```

COGNOMS, NOM:

Problema 2. (1,5 punts)

Donada la següent sentència escrita en alt nivell en llenguatge C:

```
if (((x==0) || (y!=0)) && ((y<x) || (x>0)))
            y=0;
else
            y=1;
```

Completeu el següent fragment de codi MIPS, que tradueix l'anterior sentència, escrivint en cada calaix un mnemònic d'instrucció o macro, una etiqueta, o un registre. Les variables `x` i `y` són de tipus `int` i estan inicialitzades i guardades als registres `$s0` i `$s1`, respectivament.

```
et1:        beq     $s0, $zero,  et3
et2:        beq     $s1, $zero,  et7
et3:        blt     $s1, $s0,    et5
et4:        ble     $s0, $zero,  et7
et5:        li      $s1, 0                  # y = 0
et6:        b       et8
et7:        li      $s1, 1                  # y = 1
et8:
```

COGNOMS, NOM:

Problema 3. (2,5 punts)

Donat el següent programa escrit en llenguatge C, traduïu-lo a assemblador MIPS amb el menor nombre possible de línies de codi a cada iteració del bucle.

```
short m[100][100];
main() {
int i = 0;
  while (m[i][i] > 0) {
      if (m[i][99-i] > m[i][i]) {
            m[i][99-i] = m[i][i];
      }
      i++;
  }
}
```

```
.data
m:  .space 100*100*2

.text
.globl main
main:
    la     $t1, m               # $t1: punter a m[i][i]
    addiu  $t2, $t1, 99*2       # $t2: punter a m[i][99-i]
    lh     $t3, 0($t1)
    ble    $t3, $zero, fbucle
bucle:
    lh     $t4, 0($t2)
    ble    $t4, $t3, fsi
    sh     $t3, 0($t2)
fsi:
    addiu  $t1, $t1, 101*2      # acumulem stride a $t1
    addiu  $t2, $t2, 99*2       #
    lh     $t3, 0($t1)
    bgt    $t3, $zero, bucle
fbucle:
    jr     $ra
```

COGNOMS, NOM:

Problema 4. (1,5 punts)

a) Mostreu el contingut de memòria a nivell de byte (amb les posicions d'alineament en blanc) corresponent a aquesta declaració:

```
.data                                   # comença a 0x10010000
lin1:    .half    -3, 0xff, 1
lin2:    .word    lin4
lin3:    .byte    'x'                   # el codi ascii de la 'x' és 120
lin4:    .asciiz  "xyz"
lin5:    .align   2
lin6:    .space   2
lin7:    .half    -0x7fff
```

0x10010000	**0xFD**	0x10010008	**0x0D**	0x10010010	**0x00**	0x10010018	
0x10010001	**0xFF**	0x10010009	**0x00**	0x10010011		0x10010019	
0x10010002	**0xFF**	0x1001000A	**0x01**	0x10010012		0x1001001A	
0x10010003	**0x00**	0x1001000B	**0x10**	0x10010013		0x1001001B	
0x10010004	**0x01**	0x1001000C	**0x78**	0x10010014	**0x00**	0x1001001C	
0x10010005	**0x00**	0x1001000D	**0x78**	0x10010015	**0x00**	0x1001001D	
0x10010006		0x1001000E	**0x79**	0x10010016	**0x01**	0x1001001E	
0x10010007		0x1001000F	**0x7A**	0x10010017	**0x80**	0x1001001F	

b) Indiqueu el valor final a `$t0` després d'executar aquest codi que fa referència a l'anterior declaració:

```
la    $t0, lin2
lw    $t0, 0($t0)
lhu   $t0, 1($t0)
lui   $t1, 0xA5A5
or    $t0, $t0, $t1
sra   $t0, $t0, 8
andi  $t0, $t0, -1
```

`$t0` = **0x0000A57A**

`$t0` = **0xFFA5A57A**

Nota: La darrera instrucció conté un error d'assemblatge de MARS, que decideix que hi ha sobreeiximent en representar la constant -1. Per tant, EL PROGRAMA NO ES POT EXECUTAR AL MARS. En qualsevol cas, considerem també correctes tant la solució que interpreta que s'executa aquesta darrera instrucció fent el producte lògic del register $t0 amb la màscara 0x0000FFFF (operació que es pretenia fer amb el codi de l'enunciat) com també amb la màscara 0xFFFFFFFF.

COGNOMS, NOM:

Problema 5. (2 punts)

Feu un programa que detecti si el registre `$t1` conté un valor capicua a nivel de bit, és a dir que el seu contingut es pot expressar com `$t1`: `abcd efgh ijkl mnop ponm lkji hgfe dcba`, per a qualsevol valor binari de les variables des de l'a fins la `p`. El valor final a `$t1` s'ha de codificar com CERT=1 i FALS=0.

```
main:
    li    $t0, 16
    move  $t2, $t1
bucle:
    andi  $t3, $t1, 1           # mirem bit de la dreta
    slt   $t4, $t2, $zero       # mirem bit de l'esquerra
    bne   $t3, $t4, fi_no       # si difereixen, hem acabat
    srl   $t1, $t1, 1
    sll   $t2, $t2, 1
    addiu $t0, $t0, -1
    bne   $t0, $zero, bucle
    li    $t1, 1         # és capicua
    jr    $ra
fi_no:
    li    $t1, 0         # no és capicua
    jr    $ra
```