

COGNOMS:

GRUP:

NOM:

EXAMEN PARCIAL D'EC

28 d'abril de 2020

L'examen consta de 6 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 duració de l'examen és de **120 minuts**. Les notes, la solució i el procediment de revisió es publicaran al Racó properament.

Pregunta 1. (1,5 punts)

Sigui el següent programa en C:

```
int vecs[100];
int mats[5][10];

main() {
    int i;                                /* guardat al registre $t0 */
    for (i=0; i<10; i++)
        vecs[i*10 + 4] = mats[4][9-i];
}
```

Volem traduir el bucle d'aquest programa (sols el bucle) a ensamblador MIPS, fent servir la tècnica d'accés seqüencial, tant per al recorregut de *vecs* com per al recorregut de *mats*. Determina, per a cada cas, la fórmula per calcular l'adreça del primer element del recorregut i el valor de l'stride. A continuació escriu, fent servir aquests valors, la traducció del bucle suposant que la variable *i* ocupa el registre \$t0:

mats: @ inici =

Stride =

vecs: @ inici =

Stride =

```
la    $t4, mats + 196
la    $t5, vecs + 16
li    $t0, 0
li    $t1, 10
for:
    bge $t0, $t1, fi_for
    lw  $t3, 0($t4)
    sw  $t3, 0($t5)
    addiu $t4, $t4, -4
    addiu $t5, $t5, 40
    addiu $t0, $t0, 1
    b   for
fi_for:
```

Pregunta 2. (2,25 punts)

Sigui el següent fragment de programa en C:

```
long long matl[9][5];
long long *punterl;
main() {
    int i;           /* guardat al registre $t0 */
    char q;          /* guardat al registre $t1 */
    . . .            /* sentències dels apartats */
}
```

- a) Tradueix a ensamblador MIPS la següent sentència, suposant que i ocupa el registre \$t0:
punterl = &matl[i+1][4];

```
# @matl[i+1][4] = matl + (i*5 + 1*5 + 4)*8 = matl + i*40 + 72

la    $t1, matl + 72
li    $t2, 40
mult  $t2, $t0
mflo  $t2
addu  $t1, $t1, $t2
la    $t3, punterl
sw    $t1, 0($t3)
```

- b) Tradueix a ensamblador MIPS la següent sentència, suposant que q ocupa el registre \$t1:
if ((q >= '0') || (q <= '9'))
 q = q - '0';

```
li    $t2, '0'
bge   $t1, $t2, if
li    $t2, '9'
bgt   $t1, $t2, fi_if
if:
    addiu $t1, $t1, -'0'
fi_if:
```

- c) Tradueix a ensamblador MIPS la següent sentència (compte!! punterl és un punter a un enter de doble precisió):

```
*punterl = *punterl + 8;
```

```
la    $t0, punterl
lw    $t0, 0($t0)
lw    $t1, 0($t0)
lw    $t2, 4($t0)
addiu $t3, $t1, 8
sltu  $t4, $t3, $t1
addu  $t2, $t2, $t4
sw    $t3, 0($t0)
sw    $t2, 4($t0)
```

COGNOMS:**GRUP:****NOM:****Pregunta 3. (1,75 punts)**

Donades les següents declaracions de variables globals, emmagatzemades a memòria a partir de l'adreça 0x10010000:

```
a:      .word  0x10010004
b:      .byte  0xDD
c:      .half  0x00D9
d:      .word  0xFF8406FE
e:      .half  0x0400
f:      .byte  0x40
g:      .dword 0xFFFFF00E2E05401
```

- a) Indica el contingut inicial de la memòria, representat en hexadecimal, segons el format que utilitza el simulador MARS, suposant que les posicions de memòria no inicialitzades son 0 (noteu que el simulador mostra el contingut en grups de paraules de 32 bits en hexadecimal):

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	0x10010004	0x00D900DD	0xFF8406FE	0x00400400
0x10010010	0xE2E05401	0xFFFFF000		

- b) Quin és el valor final del registre \$t1, en hexadecimal, després d'executar el següent fragment de codi?

```
la      $t0, a
lw      $t1, 0($t0)
lb      $t2, 2($t1)
sra     $t2, $t2, 2
sh      $t2, 4($t1)
lw      $t1, 4($t1)
```

\$t1 =

Pregunta 4. (1,5 punts)

Donat el següent fragment de codi:

```
li    $t1, 0x415C0403
mtc1  $t1, $f6
li    $t2, 0xBE800018
mtc1  $t2, $f4
sub.s $f8, $f6, $f4
```

- a) Indica quin és el valor final de \$f8 després d'executar el codi anterior, expressat en hexadecimal (escriu els càlculs intermedis al requadre gran, amb bona lletra)

0x415C0403 = 0100 0001 0101 1100 0000 0100 0000 0011
= + 1,10111000000010000000011 x 2³

0xBE800018 = 1011 1110 1000 0000 0000 0000 0001 1000
= - 1,00000000000000000011000 x 2⁻²

1,10111000000010000000011 x 2³
+ 0,000010000000000000000011000 x 2³
1,110000000001000000011**110** x 2³ (en negreta els bits GRS)

Abans de l'arrodoniment: 1,110000000001000000011**110** x 2³
Després de l'arrodoniment: 1,1100000000010000000100 x 2³
Codificació: 0 10000010 1100000000010000000100 = 0x41600404

Càlcul de l'error en l'arrodoniment (diferència):
0,000000000000000000000001 x 2³ = 1,0 x 2⁻²²

\$f8 =

- b) Indica si es comet error per pèrdua de precisió en l'operació de l'apartat anterior. En cas afirmatiu, calcula aquest error, i expressa'l en notació científica.

Error =

COGNOMS:

GRUP:

NOM:

Pregunta 5. (0,75 punts)

En un processador sota estudi, s'han obtingut les següents mesures:

- Freqüència de la instrucció FPSQR (arrel quadrada): 10% (sobre el total d'instruccions).
- Freqüència de les instruccions de coma flotant (excloent la instrucció FPSQR): 30%
- CPI mitjà de la instrucció FPSQR: 30
- CPI mitjà de les instruccions de coma flotant (excloent la instrucció FPSQR): 6
- CPI mitjà de la resta d'instruccions (que no són de coma flotant): 2

a) Indica la millora global de rendiment si reduïm el CPI de la instrucció FPSQR a 10.

Millora (speed-up) =

Pregunta 6. (2,25 punts)

Donada la següent declaració de funcions en C:

```
int g(int A, int *B);          /* prototip */

int func(int W[], int n, int m)
{
    int V[11];
    int x = m;

    x = x + g(n, W);
    return V[x] + x;
}
```

a) Examina el codi de la subrutina *func* i determina el nombre mínim de registres que s'hauran de salvar a la pila durant l'execució. Dibuixa el bloc d'activació de la subrutina, especificant la posició on apunta el registre \$sp un cop reservat l'espai corresponent a la pila, així com el nom de cada registre i/o variable, indicant clarament per a cada un la seva posició (desplaçament relatiu al \$sp).

b) Tradueix a ensamblador MIPS la rutina func:

Cal salvar \$ra (per la crida). Cal salvar x (en \$s0) ja que després de la crida es necessita el seu valor anterior a la crida.

```
func:
    # Reserva pila i salva context
    addiu $sp, $sp, -52
    sw     $s0, 44($sp)
    sw     $ra, 48($sp)

    # primera sentència
    move   $s0, $a2          # x = m

    # segona sentència
    move   $t0, $a0

    move   $a0, $a1          # passem n
    move   $a1, $t0          # passem W
    jal    g
    addu   $s0, $s0, $v0     # x = x + g(...)

    # tercera sentència
    sll    $t0, $s0, 2       # 4*x
    addu   $t0, $sp, $t0     # @V + 4*x
    lw     $t0, 0($t0)       # V[x]
    addu   $v0, $t0, $s0     # return V[x] + x

    # Restaura regs i allibera pila
    lw     $s0, 44($sp)
    lw     $ra, 48($sp)
    addiu  $sp, $sp, 52
    jr     $ra
```

Bloc d'activació

