Primer examen parcial **ESTRUCTURA DE COMPUTADORS** **13 de gener de 2020**

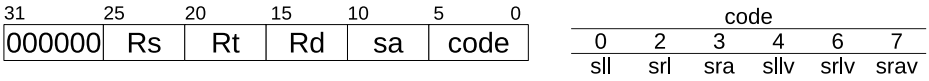
Nom i cognoms	DNI	Grup

1 **(1.5 punts)** Considereu la ruta de dades monocicle ampliada per donar suport a les instruccions de desplaçament: a l'esquerra («), lògic a la dreta ($\gg_L$) i aritmètic a la dreta ($\gg_A$). El valor desplaçat està sempre en *Rt*. Per a cada operació hi ha dues instruccions: una de desplaçament *constant*, on el nombre de bits desplaçats (*sa*, de *shift amount*, $0 \le sa \le 31$) està codificat en cinc bits dins de la instrucció, i l'altra de desplaçament *variable*, on aquest nombre està en els cinc bits inferiors del registre *Rs*. En total, són sis instruccions.

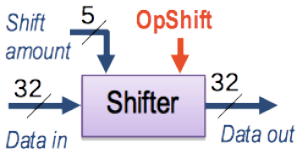
instrucció	operació
`sll rd,rt,sa`	*rd* = *rt* « *sa*
`srl rd,rt,sa`	*rd* = *rt* $\gg_L$ *sa*
`sra rd,rt,sa`	*rd* = *rt* $\gg_A$ *sa*

instrucció	operació
`sllv rd,rt,rs`	*rd* = *rt* « *rs*
`srlv rd,rt,rs`	*rd* = *rt* $\gg_L$ *rs*
`srav rd,rt,rs`	*rd* = *rt* $\gg_A$ *rs*

Noteu que en els desplaçaments constants, la ruta de dades ha d'ignorar el contingut de *Rs*. La codificació de les instruccions és aquesta:

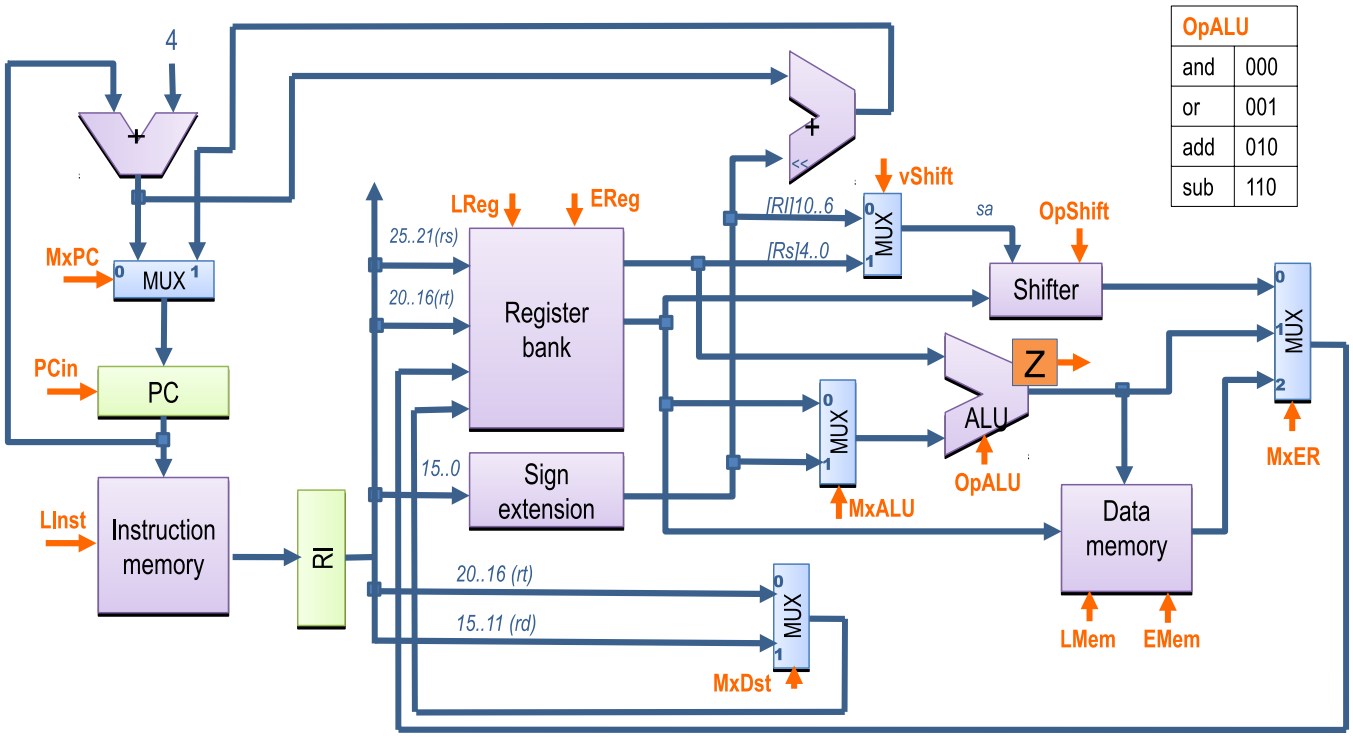


La ruta de dades té un nou component, el *shifter*, que pot fer les tres operacions seleccionades per *OpShift*.



OpShift	Operació
00	«, desplaçament a l'esquerra
10	$\gg_L$, desplaçament lògic a la dreta
11	$\gg_A$, desplaçament aritmètic a la dreta

Un nou multiplexor permet seleccionar l'origen del valor de *shift amount* mitjançant el senyal de control *vShift* (0 = desplaçament constant, 1 = desplaçament variable). I també cal ampliar el multiplexor d'escriptura de registres perquè tinga tres entrades de dades; ara, el senyal de control *MxER* tindrà dos bits. La ruta de dades queda:



Completeu la taula de descodificació de senyals de control per a cada instrucció:

Instrucció	EReg	vShift	OpShift	MxDst	MxALU	OpALU	LMem	EMem	MxER	MxPC
sll	1	0	00	1	x	xxx	0	0	00	0
sllv	1	1	00	1	x	xxx	0	0	00	0
lw	1	x	xx	0	1	010	1	0	10	0
sw	0	x	xx	x	1	010	0	1	xx	0
add	1	x	xx	1	0	010	0	0	01	0
addi	1	x	xx	0	1	010	0	0	01	0
beq	0	x	xx	x	0	110	0	0	xx	Z

2 **(1.5 punts)** Anomenem AR-1 al processador monocicle de la pregunta anterior en el què les operacions amb la memòria duren 20 ns, llegir i escriure en el banc de registres 6 ns i operar en la ALU 10 ns. Suposeu que la resta d'elements tenen un retard menyspreable.

Per augmentar la productivitat d'aquest processador se segmenta en les 5 etapes habituals (LI, DI, EX, M, ER) amb registres de segmentació és de 2ns de retard. Aquest processador és diu AR-2.

En un tercer intent de millorar les prestacions, dissenyem l'anomenada AR-3, que consisteix a supersegmentar el processador en 7 etapes subdividint en dues etapes de durada igual les etapes de memòria.

Per últim, se considera l'AR-4 que consisteix a fer l'AR-2 superescalar de dues vies.

Completeu en la Taula 1 els paràmetres que us demanen.

Ompliu la Taula 2 amb les acceleracions demandades que us permeten comparar els dissenys.

Indiqueu sempre les unitats corresponents.

TAULA 1 (0,75 punts)

	Temps de cicle	Freqüència de rellotge	Productivitat màxima
AR-1	62 ns	16,12 MHz	16,12 MIPS
AR-2	22 ns	45,45 MHz	45,45 MIPS
AR-3	12 ns	83,33 MHz	83,33 MIPS
AR-4	22 ns	45,45 MHz	90,9 MIPS

TAULA 2 (0,75 punts)

	ACCELERACIÓ MÀXIMA
AR-2 respecte d'AR-1	45,45 / 16,12 = 2,818
AR-3 respecte d'AR-2	83,33 / 45,45 = 1,833
AR-4 respecte d'AR-2	90,9 / 45,45 = 2
AR-4 respecte d'AR-3	90,9 / 83,33 = 1,09

3 **(1 punt)** En el processador segmentat amb 5 etapes de l'exercici anterior (AR-2) va a executar-se el següent fragment de codi en assemblador del MIPS R2000. Suposeu que s'insereixen cicles de parada tant per a resoldre els conflictes per dependències de dades com els de control. La latència de salt per aquest processador és d'1 cicle.

```
(1)              addi $t1, $zero, 10
(2)     bucle:   lw $t2, 0($t0)
(3)              addi $t0, $t0, -4
(4)              or $t2, $t2, $t3
(5)              addi $t2, $t2, -100
(6)              addi $t1, $t1, -1
(7)              sw $t2, 0($t4)
(8)              bne $t1, $zero, bucle
(9)              addi $t3, $t3, 1000
```

a) **(0.25 punts)** Identifiqueu els conflictes per dependència de dades que s'hi produeixen i ompliu la taula següent fent servir tantes fileres com calga:

	Registre	Número d'instrucció que hi escriu	Número d'instrucció que hi llig	Cicles de parada que cal inserir
Conflicte 1	St2	2	4	1
Conflicte 2	St2	4	5	2
Conflicte 3	St2	5	7	1
Conflicte 4	St1	6	8	0
Conflicte 5				
Conflicte 6				

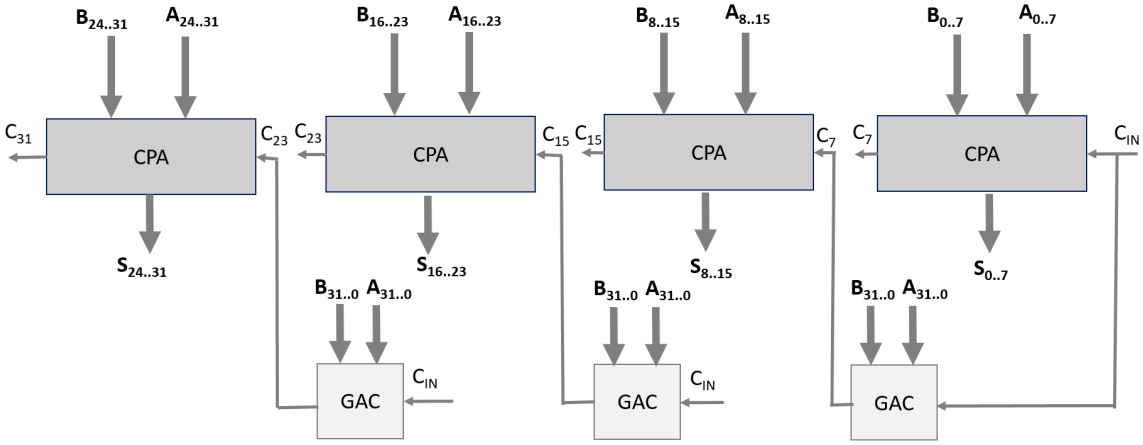
b) **(0.5 punts)** Completeu el diagrama d'instruccions-temps per a aquest fragment de codi reflectint l'**<u>última iteració del bucle</u>**

	x	x+1	x+2	x+3	x+4	x+5	x+6	x+7	x+8	x+9	x+10	x+11	x+12	x+13	x+14	x+15	x+16	x+17	x+18
lw	LI	DI	EX	M	ER														
addi		LI	DI	EX	M	ER													
or			LI	DI	DI	EX	M	ER											
addi					LI	DI	DI	DI	EX	M	ER								
addi								LI	DI	EX	M	ER							
sw									LI	DI	DI	EX	M	ER					
bne											LI	DI	EX	M	ER				
addi												LI	LI	DI	EX	M	ER		

c) **(0.25 punts)** Calculeu, per a quest codi, **tot justificant els valors**:

Nombre total d'instruccions executades (I)	1 +10*7+1 = 72
Nombre total de cicles de parada (P)	(4 dades + 1 control) * 10 = 50
Nombre total de cicles d'execució (T)	72 + 50 + 4 = 126
CPI (indiqueu els vostres càlculs)	(126-4)/72 = 1 + 50/72 = 1,694

4 **(1 punt)** La figura mostra un sumador per a dos enters (A i B) de 32 bits, basat en 4 sumadores CPA de 8 bits i 3 circuits GAC (Generadors de Transport Anticipat). Els CPA estan basats en sumadors complets FA com els estudiats en classe. Els GAC generen en paral·lel els senyals de transport C_7, C_{15} i C_{23} en funció dels valors d'entrada A i B i el transport entrant C_{IN}, amb un retard de 5ns. Tenint en compte que el retard de cada porta lògica és 1 ns, contesteu els apartats següents.



a) **(0.5 punts)** Temps necessari per a obtenir la suma d'A y B i productivitat del circuit

T = 22 ns

Productivitat = 1 /22 ns = 45,45 MOPS

b) **(0.25 punts)** Com podríeu afegir a aquest operador la detecció del desbordament en la suma d'enters amb signe? Quin seria el retard de la suma amb detecció del desbordament?

OVF = C_{31} XOR C_{30}

El retard seria el mateix

c) **(0.25 punts)** Quin és el guany de velocitat (o acceleració) d'aquest sumador respecte d'un CPA per a 32 bits?

CPA de 32 bits:: T_{CPA} = 65 ns. Productivitat = 15,38 MOPS

Acceleració = 65/22 = 45,45/15,38 = 2,95

5 (**1 punt**) En el disseny de la ALU de cert processador dotat de registres de 64 bits cal definir l'operador de multiplicació d'enters amb signe. N'hi ha dues alternatives:

- Operador seqüencial basat en l'algoritme de Booth simple. El sumador/restador té un retard de 5 ns, el desplaçament de S-HI-LO (independentment del nombre de bits a desplaçar) tarda 0.5 ns i la resta de retards és menyspreable.
- Operador seqüencial basat en l'algoritme de Booth amb recodificació per parelles. El sumador/restador té un retard de 7 ns, el desplaçament de S-HI-LO (independentment del nombre de bits a desplaçar) tarda 0.5 ns i la resta de retards és menyspreable.

Calculeu, per a ambdós operadors, la freqüència màxima a la què pot iterar l'autòmat i la productivitat resultant. En ambdós casos l'operador necessita un cicle complet de rellotge per a inicialitzar els registres.

	Algoritme de Booth simple	Algoritme de Booth amb recodificació per parelles
freqüència màxima de rellotge (MHz)	1/5.5 ns = 182	1/7.5 = 133
nombre de cicles per a operar	65	33
temps total d'operació (ns)	65 x 5.5 = 357,5	33 x 7.5 = 247,5
productivitat de l'operador (MOPS)	2,8	4

6 **(1 punt)** Escriviu una funció *areaT* en llenguatge assemblador del MIPS R2000 que calcule l'àrea d'un triangle ($A = \frac{b*a}{2}$). La base *b* del triangle i l'altura *a* son dos paràmetres reals que es passen a la funció en els registres $f10 i $f12 respectivament. L'àrea calculada torna en el registre $f0..

```
          .text 0x00400000
__start:

Dues alternatives:

areaT:    li $t0, 2
          mtc1, $t0, $f4
          cvt.s.w $f4, $f4
          mult.s $f10, $f10, $f12
          div.s $f0, $f10, $f4
          jr $ra


areaT:    li.s $f4, 2.0
          mult.s $f10, $f10, $f12
          div.s $f0, $f10, $f4
          jr $ra
```

7 (**3 punts**) Un processador MIPS R2000 té instal·lats huit mòduls idèntics de 128 MB que anomenarem DRAM0 a DRAM7. DRAM0 està ubicat en l'adreça 0x40000000, DRAM1 seguint aquest i així fins a DRAM7

a) **(0.5 punts)** Quantes paraules conté cada mòdul? Especifiqueu: nombre i noms de les línies de selecció de paraula (A_x...A_y); nombre i noms de les línies d'habilitació de byte i nombre i noms de les línies de dades.

128MB / 32 bits = 2^{25} = 32 M paraules.

25 línies de selecció de paraula: A_{26}...A_2

4 línies d'habilitació de byte: BE^*_3... BE^*_0 o DMQ^*_3...DMQ^*_0

Entrada/eixida de dades: D_{31}...D_0

b) **(0.25 punts)** Quina és la funció de selecció (a nivell baix) del mòdul DRAM0?

A_{31}	A_{30}	A_{29}	A_{28}	A_{27}	A_{26}	A_{25}	...	Funció de selecció del mòdul
0	1	0	0	0	X	X	X	A_{31}+ A^*_{30}+ A_{29}+ A_{28}+ A_{27}

c) **(0.25 punts)** Quina és l'adreça inicial del mòdul DRAM7?

0x7800 0000

d) **(0.25 punts)** Si calguera ubicar un nou mòdul DRAM8 de 256 MB en les adreces més altes del mapa de memòria, quina seria la seua adreça inicial?

A_{31}	A_{30}	A_{29}	A_{28}	A_{27}	A_{26}	A_{25}	...
1	1	1	1	X	X	X	X

L'adreça inicial seria 0xF0000000

e) **(0.25 punts)** Si el mòdul DRAM0 està format per una filera de huit xips de memòria, quina és l'organització de cada xip? Expresseu-la en la forma "$N \times w$", amb els prefixos habituals. Quantes línies de selecció de paraula tindrà el xip?

32Mx4 bits

25 línies de selecció de paraula

f) **(0.5 punts)** Els xips de memòria de DRAM8 té organització 64Mx8 bits. Cada xip conté quatre bancs de memòria. Si cada banc conté 2^{13}=8192 files, quina serà la capacitat d'una filera en KB?

Cada banc conté 16M x 8 bits.

Per tant, cada filera conté 16M/8K = 2K columnes

I cada filera conté 2 KB

g) **(0.5 punts)** Quina és l'amplada de banda de cadascun dels xips de DRAM8? Suposeu que es tracta de xips DDR treballant a 800 MHz. Quina serà l'amplada de banda del mòdul?

Xip: $800 \times 10^6 \times 2 \times 1$ B/s = 1600 MB/s

Mòdul: $800 \times 10^6 \times 2 \times 4$ B/s = 6400 MB/s

h) **(0.25 punts)** Per al mateix xip de memòria de l'apartat anterior**,** el fabricant dóna una latència de CAS (CL) de 13 cicles. Quin és el temps d'accés en ns d'una lectura quan afecta a una filera oberta?

Cicle de rellotge = 1/800 MHz = 1.25 ns

13 cicles x 1.25 ns/cicle = 16.25 ns

i) **(0.25 punts)** També segons el fabricant per al mateix xip, el mínim temps t_{RCD} és de 22 ns. Quants cicles de rellotge hauran de separar, com a mínim, les ordres ACT i RD?

22 ns / 1.25 ns/cicle = 17,6 ~ 18 cicles