Tema 5. Aritmètica d'enters i coma flotant

<5.1> **5.6.** La condición de desbordamiento (overflow) de la suma de dos números naturales *a* y *b* de 32 bits es fácil de comprobar habiendo realizado la suma *s*, pues en ese caso el valor de s (incorrecto) resulta ser menor que cualquiera de los dos sumandos. En efecto, debido al desbordamiento se cumple que $s = a+b-2^{32}$. Puesto que se cumple $b < 2^{32}$, se cumple también que $a+b-2^{32} < a$, es decir $s < a$ (análogamente se demuestra $s < b$). Basándote en esta propiedad, haz un programa que, dadas dos variables naturales de 32 bits almacenadas en $t1 y $t2, calcule si su suma ($t0 = $t1 + $t2), una vez realizada, ha producido desbordamiento (carry), en cuyo caso debe guardar un 1 en $t3, o bien un 0 en caso contrario. El programa no debe contener ninguna instrucción de salto.

```
addu    $t0, $t1, $t2
sltu    $t3, $t0, $t2       # carry = ($t1+$t2) < $t2
```

<5.1> **5.9.** Donada la següent declaració en C:

```
long long x, y;
```

Tradueix a MIPS les següents sentències: suma, resta i comparació en doble precisió:

a) `x = x + y;`

```
la      $t0, x
la      $t1, y
lw      $t2, 0($t0)         # part baixa de x
lw      $t3, 4($t0)         # part alta de x
lw      $t4, 0($t1)         # part baixa de y
lw      $t5, 4($t1)         # part alta de y

addu    $t2, $t2, $t4       # suma parts baixes
sltu    $t6, $t2, $t4       # carry de la suma: sltu
addu    $t3, $t3, $t5       # suma parts altes
addu    $t3, $t3, $t6       # suma part alta i carry

sw      $t2, 0($t0)
sw      $t3, 4($t0)
```

Nota: El carry de les parts baixes és per definició el desbordament de la suma de naturals. El problema 5.6 tracta precisament de com es calcula. La clau de l'exercici és adonar-se que cal usar sltu *i no pas* slt

b) `x = x - y;`

```
        la      $t0, x
        la      $t1, y
        lw      $t2, 0($t0)         # part baixa de x
        lw      $t3, 4($t0)         # part alta de x
        lw      $t4, 0($t1)         # part baixa de y
        lw      $t5, 4($t1)         # part alta de y

        sltu    $t6, $t2, $t4       # carry de la resta: sltu
        subu    $t2, $t2, $t4       # resta de les parts baixes
        subu    $t3, $t3, $t5       # resta de les parts altes
        subu    $t3, $t3, $t6       # restem el carry a la part alta

        sw      $t2, 0($t0)         # part baixa de x
        sw      $t3, 4($t0)         # part alta de x
```

Nota: El carry de les parts baixes és per definició el desbordament de la resta de naturals. Per calcular-lo cal simplement comparar si minuend és menor que substraend, i naturalment cal usar aquí també sltu *i no pas* slt

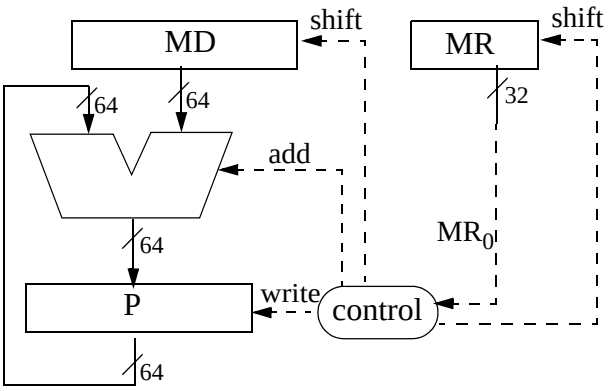
c) `if (x > y)  x = 1;`

```
        la      $t0, x
        la      $t1, y
        lw      $t2, 0($t0)
        lw      $t3, 4($t0)
        lw      $t4, 0($t1)
        lw      $t5, 4($t1)

        bgt     $t3, $t5, llavors   # compara parts altes: bgt
        blt     $t3, $t5, fi_if     # compara parts altes: blt
        bleu    $t2, $t4, fi_if     # compara parts baixes: bleu
llavors:
        li      $t7, 1
        sw      $t7, 0($t0)
        sw      $zero, 4($t0)
fi_if:
```

Nota: Cal adonar-se que quan comparem les parts altes estem comparant números "amb signe" i usem ***bgt*** *i* ***blt****: "major que" determina condició certa, i "menor que" condició falsa. Si cap de les dues comparacions no se satisfà, és perque les parts altes són iguals, i en aquest cas la condició la determinarà la part baixa. Però la part baixa no és un número en Ca2 (no té signe) i per tant no podem comparar-les amb* ***ble****. En canvi, sabem que, donades dues cadenes de bits* x *i* y*: si quan les interpretem com a enters en Ca2 es compleix que són del mateix signe i que* x>y*, llavors també es compleix que* x>y *si les interpretem com a naturals. Per tant, la comparació de les parts baixes com a naturals ens dirà quin dels dos és major: usarem* ***bleu.***

<5.2> **5.10.** Donat el següent diagrama del multiplicador seqüencial de nombres naturals (X*Y de 32 bits) estudiat a classe, el qual calcula el Producte amb 64 bits, completa l'algorisme iteratiu que en descriu el funcionament cicle a cicle:




```
MD63:32 = 0  ;
MD31:0  = X  :
MR      = Y  ;
P       = 0  ;

for (i=1; i<=32; i++)
{
      if (MR0==1)
            P = P + MD;
      MD = MD << 1;
      MR = MR >> 1;
}
```

<5.2> **5.11.** Suposant el circuit del problema 5.10, descriu els passos necessaris per a la multiplicació dels nombres naturals de 6 bits X (multiplicand) i Y (multiplicador), calculant en cada pas el valor dels registres P, MD i MR, en binari:

a) Suposant X=101000, Y=010011

iteració	Passos	P (P=0)	MD ($MD_{11:6}$=0;$MD_{5:0}$=X)	MR (MR=Y)
valor inicial		000000 000000	000000 **101000**	**010011**
1	P=P+MD MD<<=1 MR>>=1	000000 101000	000001 010000	001001
2	P=P+MD MD<<=1 MR>>=1	000001 111000	000010 100000	000100
3	No operació MD<<=1 MR>>=1	000001 111000	000101 000000	000010
4	No operació MD<<=1 MR>>=1	000001 111000	001010 000000	000001
5	P=P+MD MD<<=1 MR>>=1	001011 111000	010100 000000	000000
6	No operació MD<<=1 MR>>=1	001011 111000	101000 000000	000000

b) Suposant X=110110, Y=000100

iteració	Passos	P ($P=0$)	MD ($MD_{11:6}=0; MD_{5:0}=X$)	MR ($MR=Y$)
valor inicial		000000 000000	000000 110110	000100
1	No operació MD<<=1 MR>>=1	000000 000000	000001 101100	000010
2	No operació MD<<=1 MR>>=1	000000 000000	000011 011000	000001
3	P=P+MD MD<<=1 MR>>=1	000011 011000	000110 110000	000000
4	No operació MD<<=1 MR>>=1	000011 011000	001101 100000	000000
5	No operació MD<<=1 MR>>=1	000011 011000	011011 000000	000000
6	No operació MD<<=1 MR>>=1	000011 011000	110110 000000	000000