# Pràctica 3: Operacions i bucles amb RISC-V

## Introducció / Objectius

En aquesta pràctica treballarem l'ús de les instruccions de control de flux mitjançant la creació de bucles a través de les instruccions de salt condicionals i incondicionals. D'aquesta manera podrem ser capaços de crear qualsevol mena de bucle i condició per tal de realitzar els nostres programes. A continuació un resum de les tasques/objectius de la pràctica:

- Entendre les diferents instruccions de salt i les seves sigles.
- Entendre la diferència entre instrucció de salt condicional i incondicional.
- Aprendre a realitzar bucles en llenguatge assemblador.
- Solidificar els conceptes apressos sobre registres en les pràctiques anteriors.
- Guanyar fluïdesa en la generació d'algorismes.

## Exercicis guiats

### Instruccions de salt:

| Instrucció | Operació | Equivalent |
|---|---|---|
| beq rs, rt, destí | Salta a "destí" si el registre rs = rt | |
| bne rs, rt, destí | Salta a "destí" si el registre rs ≠ rt | |
| blt rs, rt, destí | Salta a "destí" si el registre rs < rt | |
| bge rs, rt, destí | Salta a "destí" si el registre rs ≥ rt | |
| bltu rs, rt, destí | Salta a "destí" si el registre rs < rt (*) | |
| bgeu rs, rt, destí | Salta a "destí" si el registre rs ≥ rt (*) | |
| beqz rs, destí | Salta a "destí" si el registre rs = 0 | beq rs, zero, destí |
| bnez rs, destí | Salta a "destí" si el registre rs ≠ 0 | bne rs, zero, destí |
| blez rs, destí | Salta a "destí" si el registre rs ≤ 0 | bge zero, rt, destí |
| bgez rs, destí | Salta a "destí" si el registre rs ≥ 0 | bge rs, zero, destí |
| bltz rs, destí | Salta a "destí" si el registre rs < 0 | blt rs, zero, destí |
| bgtz rs, destí | Salta a "destí" si el registre rs > 0 | blt zero, rs, destí |
| bgt rs, rt, destí | Salta a "destí" si el registre rs > rt | blt rt, rs, destí |
| ble rs, rt, destí | Salta a "destí" si el registre rs ≤ rt | bge rt, rs, destí |
| bgtu rs, rt, destí | Salta a "destí" si el registre rs > rt (*) | bltu rt, rs, destí |
| bleu rs, rt, destí | Salta a "destí" si el registre rs ≤ rt (*) | bgeu rt, rs, destí |
| j destí | Salta a "destí" incondicionalment | |

* Comparacions sense complement a dos, considerant només números positius.

**Problema 1:**

Els salts ens permeten executar diferents blocs de codi i així podem implementar estructures de control de flux.

Exemple **if**: distància entre dos números enters

Per calcular la distància entre dos números enters, farem la resta i llavors el valor absolut:

$$|a - b|$$

Exemples:

$$|5 - (-6)| = 11$$

o

$$|-6 - 5| = 11$$

El programa següent en C calcula aquesta distància.

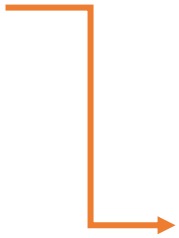
**Alt nivell (C)**

```
int a = 5;
int b = -6;
int resultat;

if (a >= b)
 resultat = a - b;
else
 resultat = b - a;
```

Com es faria en RISC V?

**RISC V**

```
.data
a: .word 5
b: .word -6
resultat: .word 0
.text
        la a0, a
        lw a1, 0(a0)
        lw a2, 4(a0)
        # condició
cert:
        # branca certa
fals:
        # branca falsa
end:
        sw a3, 8(a0)
```



```
1  .data
2  a:  .word 5
3  b:  .word -6
4  resultat:  .word 0
5
6  .text
7     la a0, a
8     lw a1, 0(a0)
9     lw a2, 4(a0)
10
11    # condició: a >= b -> branca certa
12    # a < b -> branca falsa
13    blt a1, a2, fals
14
15 cert:
16    # branca certa
17    sub a3, a1, a2
18    j end
19
20 fals:
21    # branca falsa
22    sub a3, a2, a1
23
24 end:
25    sw a3, 8(a0)
```

Resultat del programa al executar-ho amb l'exemple amb els nombres 5 i -6:

| | | |
|---|---|---|
| x11 | a1 | 5 |
| x12 | a2 | -6 |
| x13 | a3 | 11 |

| | | | | | |
|---|---|---|---|---|---|
| 0x10000008 | 11 | 11 | 0 | 0 | 0 |
| 0x10000004 | -6 | 250 | 255 | 255 | 255 |
| 0x10000000 | 5 | 5 | 0 | 0 | 0 |

Preguntes a resoldre:

1. **Quines instruccions de salt condicional hem fet servir? En quin cas salten?**
   Només hem fet ús d'una instrucció de salt condicional i aquesta ha estat: ***blt a1, a2, fals*** (salta a *fals* si a1 <= a2).
2. **Què fan les instruccions de salt condicional quan la condició no es compleix?**
   Si la condició no es compleix no es realitza el salt. Per tant el que es fa es continuar amb la següent instrucció seguint l'ordre habitual d'execució del codi, per ordre en que apareixen.
3. **Per què hem utilitzat les instruccions de salt incondicional?**
   La instrucció de salt incondicional que hem fet servir ha estat: ***j end*** (salta a *end* incondicionalment). L'hem utilitzat per saltar al final del programa sense passar per la branca falsa en cas que la instrucció de salt condicional utilitzada (*blt a1, a2, fals*) no es compleixi. Ja que una vegada s'executa la branca certa, seguint l'ordre habitual de les instruccions s'executaria després la branca falsa, i per tal d'evitar això fem un salt incondicional.

**Problema 2:**

Exemple **while**: *Fibonacci*

La successió de Fibonacci és una successió matemàtica de nombres naturals tal que cada un dels seus termes es igual a la suma dels dos anteriors.

$F_n = F_{n-1} + F_{n-2}$

sempre es parteix dels valors inicials

$F_0 = 0$, $F_1 = 1$

i es a partir del elements $F_0$ i $F_1$ que es genera la resta d'elements. Exemples:

$F_2 = 1$

$F_3 = 2$

$F_4 = 3$

...

$F_9 = 34$

$F_{10} = 55$

El programa següent calcula el terme de la sèrie de Fibonacci indicat per la variable "comptador" ($F_{comptador}$), i el desa a la variable "resultat".

Alt nivell (C)

```
int comptador = 10;
int resultat;

int a = 0;
int b = 1;
while (comptador > 0) {
  int t = a + b;
  a = b;
  b = t;
  comptador--;
}
resultat = a;
```

Com es faria en RISC V?

**RISC V**

```
.data
comptador: .word 10
resultat: .word 0
.text
        la a0, comptador
        lw a0, 0(a0)
        addi a1, zero, 0
        addi a2, zero, 1
loop:
        # condició
        # cos del bucle
end:
        la a0, resultat
        sw a1, 0(a0)
```

```
1  .data
2  comptador: .word 10
3  resultat: .word 0
4
5  .text
6     la a0, comptador
7     lw a0, 0(a0)
8     addi a1, zero, 0
9     addi a2, zero, 1
10
11 loop:
12    # condició: comptador > 0 -> em quedo en el loop
13    # comptador = 0 -> surto del loop
14    beqz a0, end
15    #cos del bucle
16    add a3, a1, a2
17    mv a1, a2
18    mv a2, a3
19    addi a0, a0, -1
20    j loop
21
22 end:
23    la a0, resultat
24    sw a1, 0(a0)
```

Resultat del programa al executar-ho *comptador = 10 (hauríem d'esperar resultat = 55)*:

| x11 | a1 | 55 |
|---|---|---|

| | | |
|---|---|---|
| 0x10000004 | 55 | 55 |
| 0x10000000 | 10 | 10 |

Preguntes a resoldre:

1. **Quin tipus d'estructura de control de flux indica normalment un salt cap enrere?**
   Un bucle, per exemple un bucle ***while*** com el que hem fet servir en la realització del programa per a calcular la sèrie de *Fibonacci*.
2. **Omple la taula següent executant el programa:**

| VALOR INICIAL | VALOR A "RESULTAT" | #CICLES | #INSTRUCCIONS |
|---|---|---|---|
| $F_0$ | 0 | 15 | 9 |
| $F_1$ | 1 | 23 | 15 |
| $F_2$ | 1 | 31 | 21 |

| | | | |
|---|---|---|---|
| **$F_3$** | 2 | 39 | 27 |
| **$F_4$** | 3 | 47 | 33 |
| **$F_5$** | 5 | 55 | 39 |
| **$F_6$** | 8 | 63 | 45 |
| **$F_{25}$** | 75025 | 215 | 159 |
| **$F_{46}$** | 1836311903 | 383 | 285 |
| **$F_{47}$** | -1323752223 | 391 | 291 |

**3. Què passa amb el resultat $F_{47}$?**

Surt un nombre negatiu ja que el resultat de fer *Fibonacci* de 47 és $2{,}971215073 \cdot 10^9$ i el rang de nombres màxim que pot mostrar un *.word* són $2^{31} = 2{,}147483648 \cdot 10^9$.

**Problema 3:**

Per fer a casa: implementa les parts que falten del programa assemblador.

Exemple ***while amb if***: màxim comú divisor.

El màxim comú divisor (mcd) de dos o més nombres enters positius és el major divisor possible de tots ells.

mcd(a, b)

Exemples:

mcd(252, 105) = 21

mcd(13, 31) = 1

A continuació hi ha la implementació en C usant l'algorisme d'Euclides. Dins d'un bucle "***while***", a cada iteració es fa una comparació en un "***if***".

Alt nivell (C)

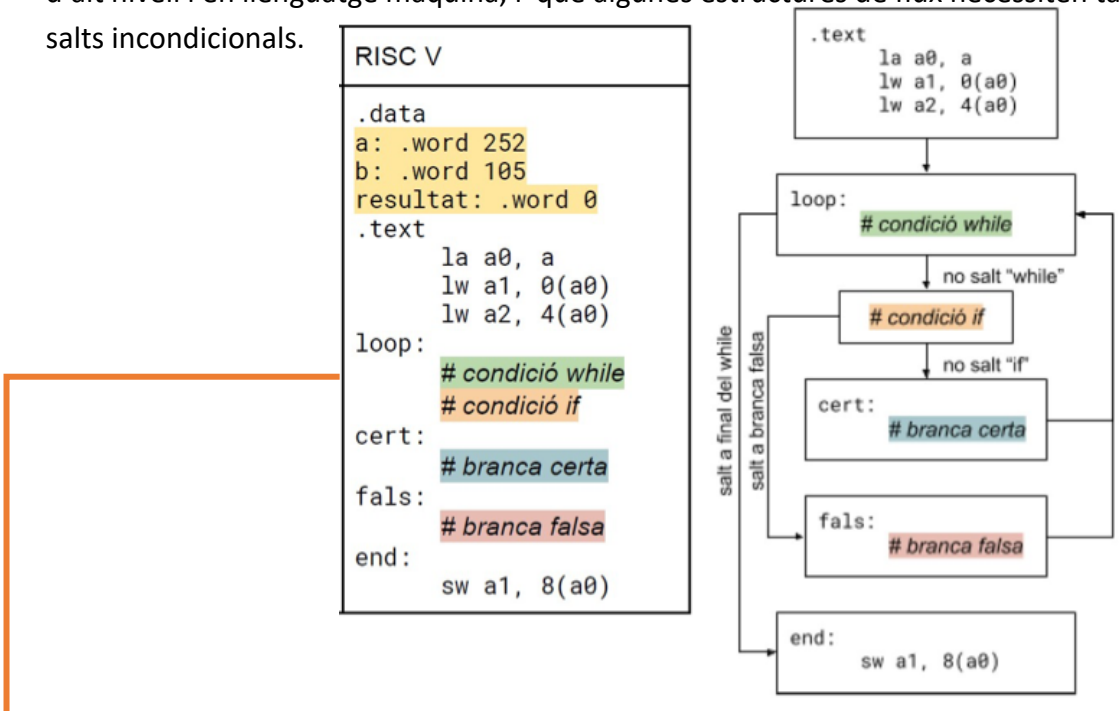
```
int a = 252;
int b = 105;
int resultat;

while (a != b) {
  if (a > b)
    a = a - b;
  else
    b = b - a;
}
resultat = a;
```

Com es faria en RISC V?

Tingues en compte el flux del programa assemblador, i el que hem explicat en els exercicis anteriors sobre com avaluem de forma diferent les condicions en llenguatge d'alt nivell i en llenguatge màquina, i que algunes estructures de flux necessiten també salts incondicionals.

RISC V

```
.data
a: .word 252
b: .word 105
resultat: .word 0
.text
      la a0, a
      lw a1, 0(a0)
      lw a2, 4(a0)
loop:
      # condició while
      # condició if
cert:
      # branca certa
fals:
      # branca falsa
end:
      sw a1, 8(a0)
```



```
.data
a: .word 252
b: .word 105
resultat: .word 0

.text
   la a0, a
   lw a1, 0(a0)
   lw a2, 4(a0)

loop:
   beq a1, a2, end
   ble a1, a2, fals

cert:
   sub a1, a1, a2
   j loop

fals:
   sub a2, a2, a1
   j loop

end:
   sw a1, 8(a0)
```

Resultat del programa al executar-ho *amb a = 252, b = 105*, *(hauríem d'esperar resultat = 21):*

| | | |
|---|---|---|
| x11 | a1 | 21 |

Pregunta a resoldre:

1. **Descriu el programa que has implementat. Quins salts has usat? Quins són condicionals i quins són incondicionals, i per què?**
   Per convertir el programa d'alt nivell en C a llenguatge assemblador RISC-V ho he fet de la següent manera:
   - Primerament hem creat dues variables *.word* **a** i **b**, amb les quals treballarem i operarem fins a trobar el seu màxim comú divisor, el qual emmagatzemem a la variable '**resultat**', la qual sempre inicialitzarem amb valor 0.
   - Una cop creades les variables principals i les adreces i registres corresponents crearem les condicions principals en la que es basarà el càlcul per trobar el *mcd*. Amb la instrucció ***beq*** indicarem que quan el valor dels registres **a1** i **a2** siguin iguals farem un salt condicional al final del programa, ja que si els dos valors són iguals ens indica que ja hem trobat el seu màxim comú divisor (els valors que es trobarien emmagatzemats en aquell moment en **a1** i **a2**).
   - En cas de que aquesta condició no es complís passarem a analitzar la següent, ***ble***, que ens comprovarà si el valor del registre **a1** és més petit o igual al valor del registre **a2**, i en cas de complir-se saltaria a la branca falsa, la qual realitzaria la resta del valor del registre **a2** menys el valor del registre **a1** (ha de ser en aquest ordre ja que el valor del registre **a2** > que

el valor del registre **a1**) i el guarda en **a2**. I una vegada finalitzada l'operació farem un salt incondicional al principi del nostre ***loop*** per tornar a analitzar els valors dels registres **a1** i **a2**.

- En cas que tampoc es compleixi aquesta última instrucció (és a dir, valor del registre **a1** <= valor registre **a2**) entrem a la branca certa, però sense necessitat de fer cap salt, ja que al no executar-se els salts condicionals el programa simplement segueix executant-se en l'ordre d'instruccions establert, executant així les instruccions que correspondrien a la branca certa.
- Com que només s'arriben a executar aquestes instruccions quan el valor del registre **a1** > valor del registre **a2**, ara el que realitzarem serà restar el valor del registre **a1** menys el valor del registre **a2** i guardarem el resultat en **a2**, i tornarem de nou mitjançant un salt incondicional al principi del ***loop***.
- D'aquesta manera el programa anirà operant i realitzant restes (seguint l'algorisme d'Euclides) fins que en un moment donat els valor dels registres **a1** i **a2** coincideixin, moment en què haurem trobat el seu *mcd* i sortirem finalment del bucle.

## Conclusions

En aquesta pràctica hem assolit un cert nivell fonamental de les instruccions de salt, tant condicionals com incondicionals, amb les quals podem generar bucles i així crear estructures de control de flux. A continuació un resum del que hem après:

- Hem après el significat de totes les sigles de les instruccions de salt.
- Hem après la diferència a l'hora d'establir condicions entre llenguatge d'alt nivell i llenguatge assemblador.
- Hem posat en pràctica els conceptes apresos en diversos exercicis on per resoldre'ls és necessari la utilització de bucles amb salts condicionals i incondicionals.