# Informe de la pràctica 4: Microinstruccions en SiMR

## Introducció

Visualitzar els passos que fa el microprocessador per executar una instrucció, és a dir, ampliar més a fons el SiMR per aprendre la unitat més petita de la màquina rudimentària, mitjançant els ja apresos bucles i algoritmes diversos.

## Base Teòrica

- Per realitzar la pràctica es fa servir el simulador SIMR.

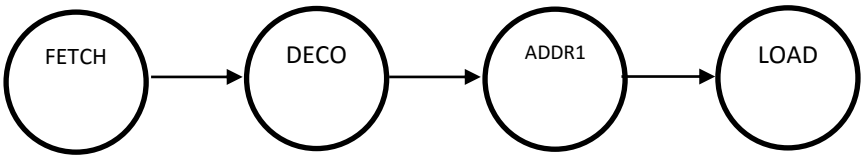
- Primer de tot s'observa com funcionem els número de cicles, és ad dir, les accions que tenen lloc en cada cicle.
- Aquestes microinstruccions, en conjunt, s'anomenen instruccions i en el SIMR tenen una longitud de 16 bits i una execució que va des de 3 fins a 5 cicles.

## Preguntes

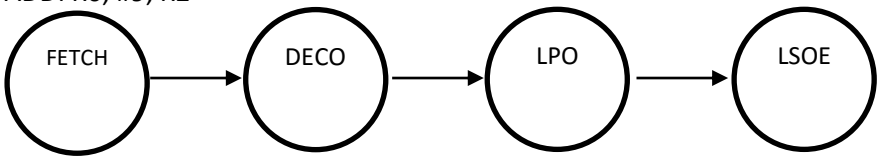
**1.** Explica detalladament la pràctica realitzada. Fes els diagrames necessaris per entendre i mostrar el cicle d'execució dels diferents tipus d'instruccions.

```
valorDada: .dw 7

guardaResultat: .rw 1

.begin start

start:    LOAD valorDada(R0), R1

    a.    ADDI R0, #9, R2

          ADD R0, R0, R3

loop:     ADD R2, R3, R3

SUBI R7, #1,R7

BG loop

STORE R3, guardaResultat(R0)

.end
```

1. LOAD valorDada(R0), R1



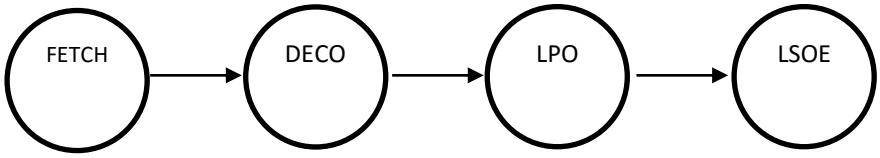

2. ADDI R0, #9, R2




LPO: Carrega el primer operand(R0)

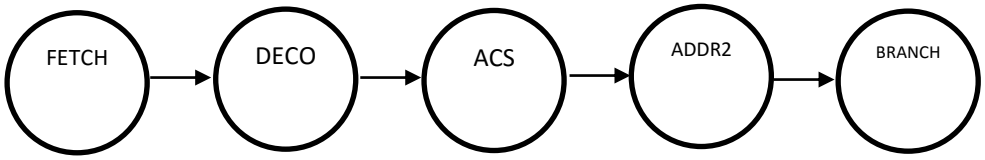
LSOE: Carrega el segon operand(#9) i executes

3. ADD R0, R0, R3



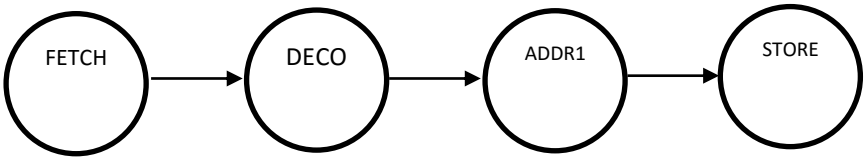

A LPO es carrega el primer operand(R0) i a LSOE es carrega el segon operand(R0) i executes(R3)

4. BG loop




ACS: Acumulador a ADDR2 compares i a BRANC Saltes

5. STORE R3, guardaResultat(R0)




Fas ADDR1 amb el “guardaResultat”, i posteriorment ho guardes a STORE

**Preguntes**

1. *Quan es produeix el salt, quants cicles triga en executar-se la instrucció BG loop?*

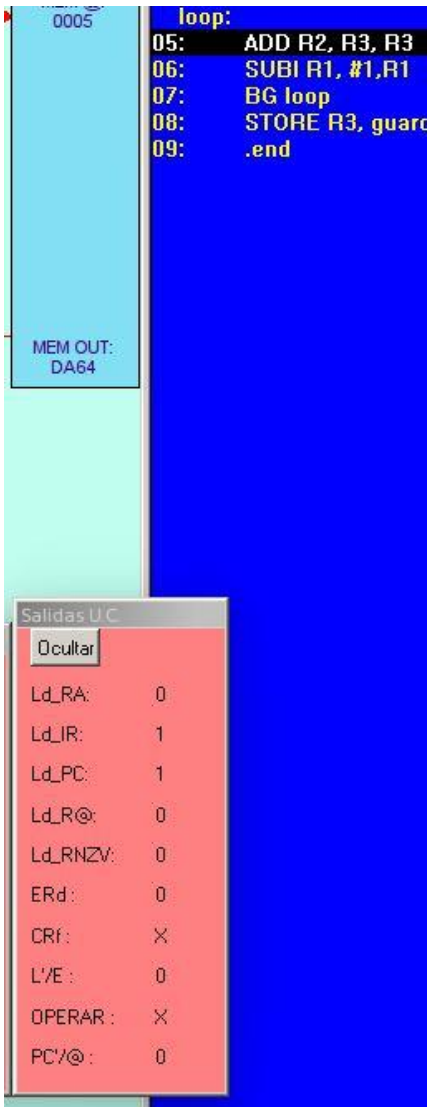
   5 cicles, FETCH, DECO, ACS, ADDR2 i BRANCH (saltar)

2. *Quan NO es produeix el salt, quants cicles triga en executar-se la instrucció BG loop?*

   3 cicles, FETCH, DECO i ACS.

3. *Que bits del bus de control s'activen en el primer cicle de la instrucció ADD R2, R3, R3*

   S'activen els bits de control Z = 1 i N= 0. I Ld_A i Ld_R són igual a 1, Ld_PC a 1, Ld_R@, Ld_RNZV i ERd a 0, CRf X (indiferent), L'/E a zero, OPERAR indiferent i per últim PC'/@ a zero:



4. *És igual la resposta del processador en el primer cicle de la instrucció ADD R2,R3,R3 que en el primer cicle de la instrucció LOAD valorDada(R0),R1?*

   No, s'activen els bits de control Z = 0 i N= 0. A més, Ld_RZ i Ld_Rn són igual a 1, a diferencia de la instrucció ADD R2, R3, R3

5. *Quan es produeix el salt (instrucció BG loop) , quina entrada del multiplexor s'activa com a sortida per apuntar a una determinada posició de la memòria? A quina posició apunta?*

   Per veure-ho hem d'executar la instrucció BG loop en cicles, i veure en el diagrama del nostre processador que les dues entrades del multiplexor (R@ i PC) van canviant segons si salta o no.

   En aquest cas, s'activa l'entrada Ld_IR del multiplexor que és igual 1, per tant, agafarà les dades de IR que són B805 que és la que hi ha en aquest moment, i apunta a la posició de memòria 5. També s'activa Ld_PC que apunta la posició a 05 de les instruccions.

6. *Quan no es produeix el salt (instrucció BG loop) , quina entrada del multiplexor s'activa com a sortida per apuntar a una determinada posició de la memòria? A quina posició apunta?*

   Mentre que en aquest segon cas s'activa la entrada LD_A i la entrada Ld_R que són igual a 1. RA apunta a la posició 0000 i R@a apunta a la posició 05.

## Conclusions

He visualitzat els passos que fa el microprocessador per executar una instrucció, observar com funcionem els número de cicles i entendre'ls.

```
Dades: .DW 3083,17 ;valors a carregar a les posicions de memòria 1 i 2.
.begin inici ;directiva d'inici de programa
inici:
LOAD 0(R0), R1 ;inicialitza R1 al contingut de la posició de memòria 0.
LOAD 1(R0), R2 ;carrega a R2 el contingut de la posició de memòria 1.
loop: ;while
ADD R7, R1, R7 ;Suma R7 + R1 i guarda-ho en R7
SUBI R2,#1, R2 ;Resta el contingut de R2 menys 1
BG loop ;si la operació anterior dona positiu, salta a la posició de memòria "loop"
STORE R7, 0(R2) ;Guarda el contingut de R7 en la posició de memòria del contingut de R2.
.end
```

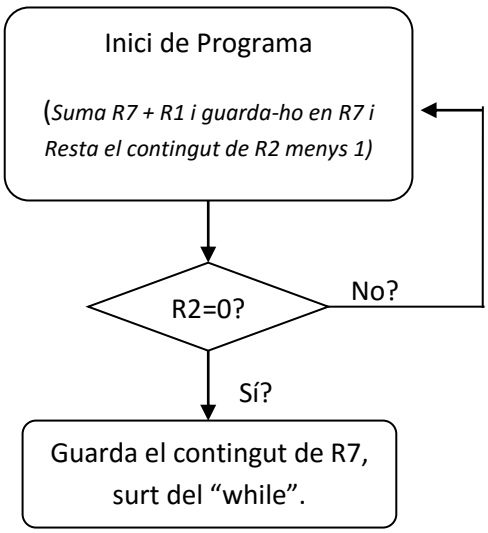



*Diagrama de flux 1: Programa de l'exercici 2*

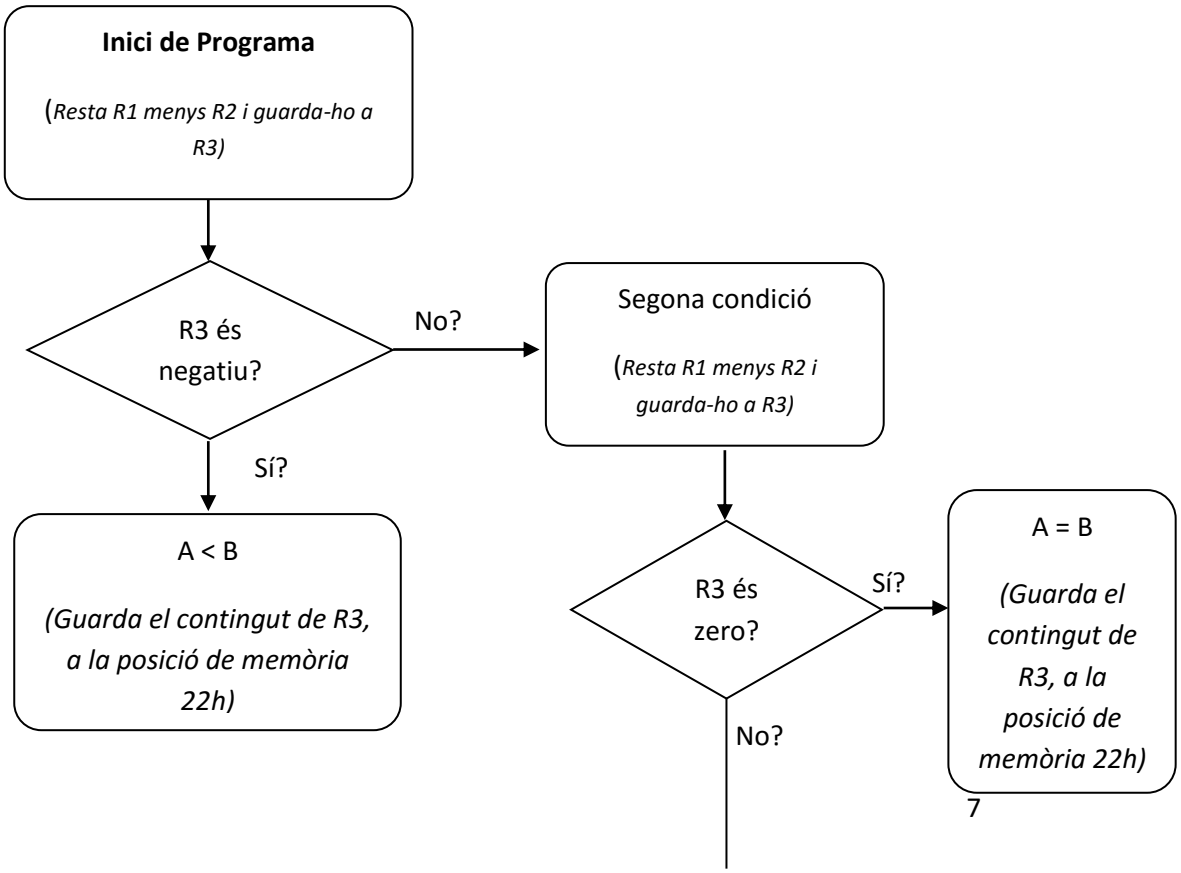
***1.*** *Ens demanen calcular un algoritme que ens faci el següent: Donades dues entrades emmagatzemades en les posicions de memòria A i B, fer la comparació. Si A>B calculem la suma. Si B>A fem la diferència (B-A) i si són iguals que finalitzi l'algoritme.*

Per fer-ho, està clar que s'haurà de fer us de les instruccions de tipus salt condicional i incondicional. Les nostres condicions en aquest cas són 3: B més gran que A (BL) A i B iguals (BEQ) i A més gran que B.

El que faig és el següent: carrego les dades a R1 i R2, les resta i les guarda a R3. Si el resultat és negatiu salta a guardar R3 a la posició de memòria 22h i acaba, sinó, torno a fer la mateixa resta, per comprovar si són iguals. En conseqüència, si el resultat dona zero, salta al final i acaba, sinó, suma R1 i R2 i ho guarda a R3, ja que voldrà dir que A és més gran que B. Després, guarda-ho a la posició de memòria 22h.

```
Dades: .DW 4,3
```

```
.begin inici ;directiva d'inici de programa
inici:
        LOAD 0(R0), R1 ;carrega el primer valor a R1
        LOAD 1(R0), R2 ;carrega el segon valor a R2
        SUB R1,R2,R3 ;resta R1 menys R2
        BL guarda ;Si el resultat de la operació anterior és negatiu, ja he acabat (b és mes gran que a)
        SUB R1,R2,R3 ;Torna a restar R1 i R2 per comprovar si són iguals
        BEQ fi ;Si la operació anterior ha donat zero, ja he acabat (són iguals)
        ADD R1,R2,R3 ;Si he arribat fins aquí vol dir que A és més gran que B. Suma.
        BR guarda ;Salt incondicional a guardar el resultat
guarda:
        STORE R3, 22(R0) ;guarda el resultat en la posició de memòria 22h.
fi:
        .end
```



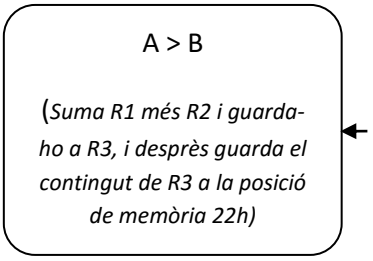



*Diagrama de flux 2: Algoritme de l'exercici 3*

**Preguntes**

1) Exercici 1
    a) Al acabar el programa, R5 val 0009h, R6 val 0008h i R3 val 0000h.
2) Exercici 2
    a) R7 val CCBBh, que seria 1100 1100 1011 1011 en binari i 12249977906120007 85 en decimal.
    b) El programa s'executa 17 vegades
    c) Es guarda en la posició de memòria 0000h, ja que R2 val zero quan el programa acaba.
3) Exercici 3
    a) Explicat a dalt.
    b) Implementat a dalt.

**Conclusions**

En aquesta pràctica he après els coneixements mínims d'algorismes en llenguatge màquina amb les instruccions de tipus de salt de les que disposo, de manera que a la vegada he pres contacte amb els registres de la meva CPU i els seus valors.

- He aconseguit familiaritzar-me amb el funcionament dels registres de la CPU
- He aconseguit que l'algoritme realitzés el demanat.