

Pràctica 4: Microinstruccions en Ripes i SiMR

(1 sessió)

Objectiu

L'objectiu de la pràctica és visualitzar els diferents passos que ha de fer un microprocessador per tal d'executar una instrucció.

Introducció

El número de cicles dividit per la freqüència de funcionament del processador ens defineix el temps d'execució de la instrucció. Les diferents accions que es realitzen en cada cicle forma el que es coneixen com microinstruccions.

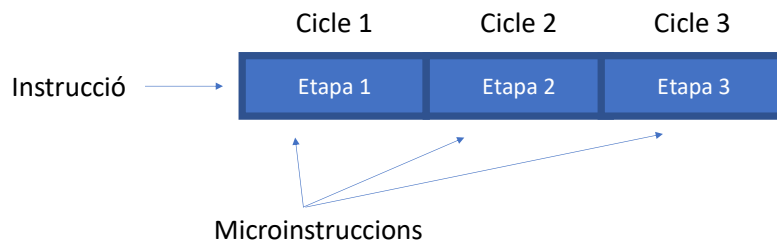
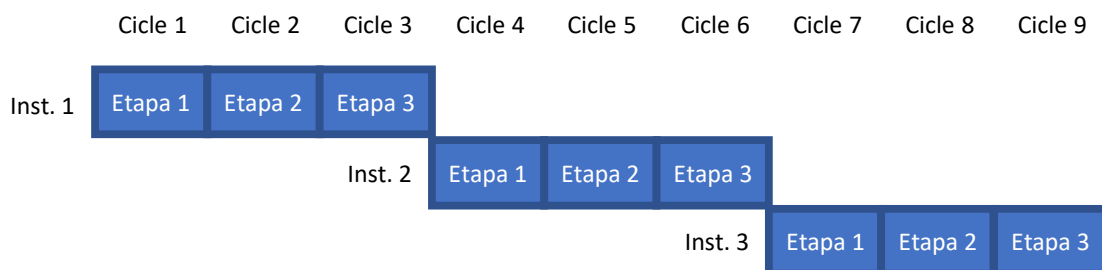


Figura 1. Concepte de microinstrucció

Com ja vàrem veure a l'apartat *Estructura bàsica de la CPU* d'aquest guió de pràctiques de l'assignatura, hi ha processadors que han d'executar totes les etapes d'una instrucció abans d'executar la següent, però també n'hi ha d'altres que a costa de multiplicar recursos de hardware poden executar diferents etapes d'una instrucció com a una cadena de muntatge, o *pipeline*.

a) Sense Pipeline



b) Amb Pipeline

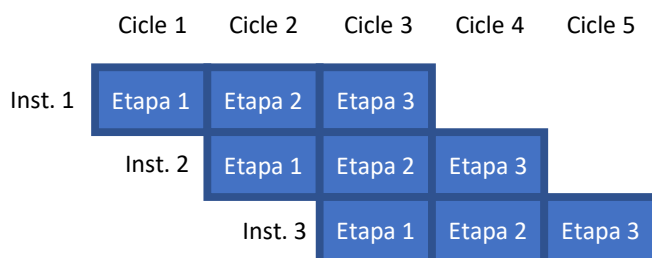


Figura 2. Cronograma d'execució d'instruccions en un processador a) sense *pipeline* i b) amb *pipeline*. El temps avança seguint un eix imaginari horitzontal de esquerra a dreta. Pel que fa a les instruccions, aquestes s'incrementen seguint un eix imaginari vertical que va de dalt a baix.

En ciència de computadors, el *pipeline* RISC es una solució de microarquitectura que té com a propòsit la execució d'una instrucció per circle de rellotge. Aquest *pipeline* executa una instrucció en cinc etapes: Instruction Fetch, Instruction Decode, Execute, Memory Access i Writeback. A continuació s'expliquen les accions que realitza el processador en cadascuna d'aquestes etapes:

- **Instruction Fetch (IF):** Llegeix la següent instrucció de memòria i incrementa el comptador de programa.
- **Instruction Decode (ID):** Decodifica la instrucció. Llegeix els registres d'operands i computa direccions de salt (si escau).
- **Execute(EX):** Executa una operació aritmètico-lògica o realitza un salt.
- **Memory Access (MEM):** Realitza accessos a memòria de lectura o escriptura.
- **Writeback(WB):** Escriu els resultats de les operacions a un registre.

Alguns dels processadors que comparteixen aquesta microarquitectura són: MIPS (Microprocessor without Interlocked Pipelined Stages), SPARC (Scalable Processor Architecture), Motorola 88000 (o m88k) i, es clar, el RISC-V (Reduced Instruction Set Computer). A la següent figura podem veure el típic pipeline de cinc etapes d'una màquina RISC.

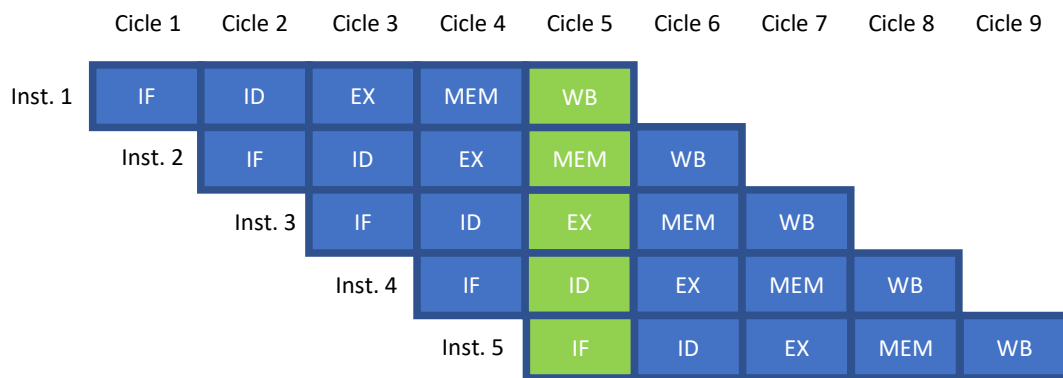


Figura 3. Pipeline de cinc etapes bàsic a una màquina RISC. Veiem que al cinquè cicle de rellotge (columna verda) la primera instrucció s'acaba d'executar, la segona instrucció es troba a l'etapa IF, la tercera a EX, la quarta a ID, i la cinquena a IF. Noteu que si no hi haguèss esperes (stall(s) deguts a un hazard), a partir del cinquè cicle cada instrucció s'executaria en un cicle de rellotge.

Exercici guiat

Haurem d'executar els següents codis, cicle a cicle (microinstrucció a microinstrucció):

Codi 1

```
.data
resultat: .word 0
.text
main:
add a3, zero, zero
add a7, zero, zero
addi a2, zero, 4
add a3, a2, a3
addi a7, a7, -1
bgt a7, zero, salta
salta:
la a0, resultat
sw a3, 0(a0)
```

Codi 2

```
.data
resultat: .word 0
.text
main:
add a3, zero, zero
add a7, zero, zero
addi a2, zero, 4
add a3, a2, a3
addi a7, a7, -1
la a0, resultat
sw a3, 0(a0)
```

- 1) Abans d'executar els codis tracta d'esbrinar la seva funcionalitat. Faran el mateix? Creus que trigaran el mateix nombre de cicles en executar-se?
- 2) Ves a la finestra del Ripes dedicada al procesador (Processor). Busca entre les opcions del Simulator control, la Pipeline table.
- 3) Executa els dos codis cicle a cicle fixant-te com les instruccions van passant per les diferents etapes del pipeline. Compta els nombre de cicles que es necessiten per executar cadascun dels codis i compara les Pipeline tables.
- 4) Què signifiquen el signes '-' que apareixen a les Pipeline tables?
- 5) Com afectaria al nombre total de cicles d'execució el següent canvi en el codi:

Codi 1

```
.data
Resultat: .word 0
.text
main:
add a3, zero, zero
add a7, zero, zero
addi a2, zero, 4
add a3, a2, a3
addi a7, a7, -1
bgt a7, zero, salta
salta:
la a0, Resultat
sw a3, 0(a0)
```



Codi 3

```
.data
Resultat: .word 0
.text
main:
la a0, Resultat
add a3, zero, zero
add a7, zero, zero
addi a2, zero, 4
add a3, a2, a3
addi a7, a7, -1
sw a3, 0(a0)
```

Realització de la pràctica

Ripes: Executa el següent programa microinstrucció a microinstrucció:

Codi 4

```
.data
valorDada: .word 2
guardaResultat: .word 0
.text
main:
lw a7, valorDada
addi a2, zero, 9
add a3, zero, zero
loop:
add a3, a2, a3
addi a7, a7, -1
bgt a7, zero, loop
la a0, guardaResultat
sw a3, 0(a0)
```

SiMR: Tradueix el codi anterior de ensamblador de RISC-V a ensamblador de màquina rudimentària. Per fer la conversió, utilitza el següent el conveni de selecció de registres de la Taula 1. Nota que no es necessari cap registre equivalent a a0 per realitzar aquest codi en SiMR. Això es simbolitza a la taula amb una 'x'.

Simulador	Registres				
Ripes	a0	a2	a3	a7	zero
SiMR	x	R2	R3	R7	R0

Taula 1. Conveni de selecció de registres per passar el codi de Ripes a SiMR.

Executa el codi que facis en SiMR microinstrucció a microinstrucció.

Informe

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

Preguntes sobre el simulador RIPES:

Per resoldre aquestes qüestions es necessari mirar l'estat del pipeline o utilitzar la *Pipeline table*:

- 1) Quin es l'estat de cadascuna de les cinc etapes del pipeline al cicle 6? I al 8?
- 2) Quins senyals de control s'activen en el cicle 4? A quines instruccions del codi corresponen?
- 3) Quins són els valors a les sortides dels multiplexors assenyalats a la figura al cicle 7:

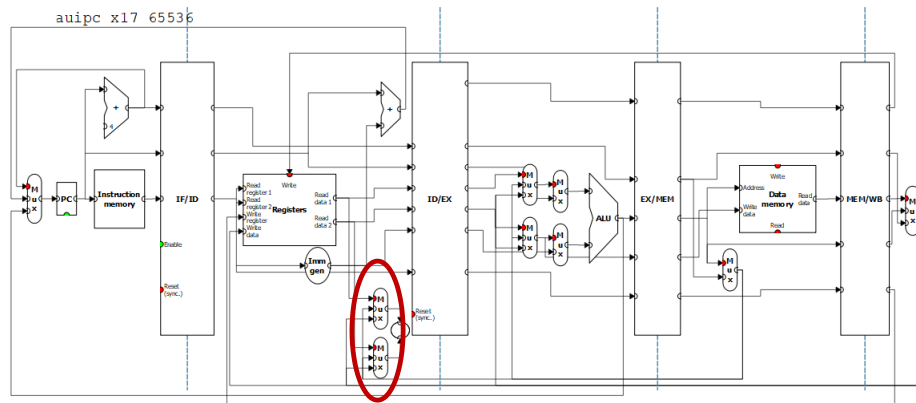


Figura 4. Detall de les etapes del RISC-V en el simulador RIPES.
L'el·lipse vermella assenyalta els multiplexors que s'utilitzen a la qüestió 3.

- 4) Perquè els valors apareixen en aquest ordre?
- 5) Llegeix amb cura la part del codi amb que s'implementa el loop. Tenint en compte el que has vist a la qüestió 3), justifica perquè el pipeline al cicle 9 presenta aquest estat:

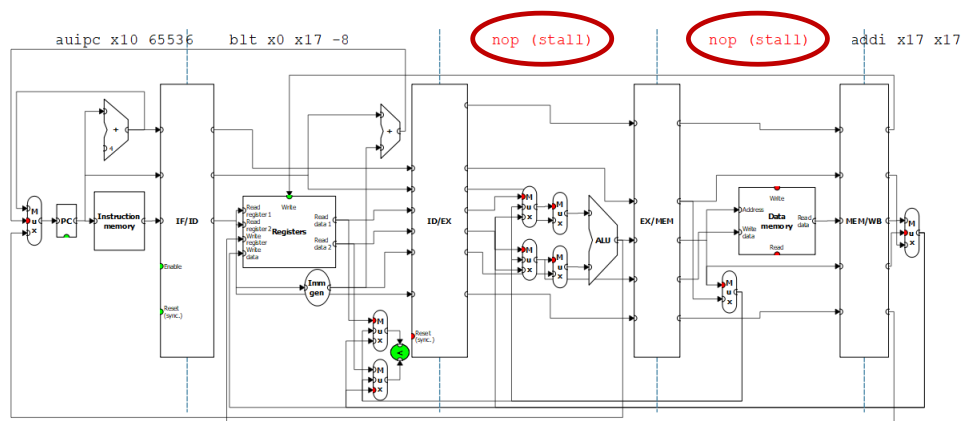


Figura 5. Estat del pipeline al cicle 9.

- 6) Quan NO es produeix el salt, quants cicles triga en executar-se la instrucció `bgt a7, zero, loop`?
- 7) Quan s'està executant la instrucció de salt, però el salt NO es produeix, a quina posició apunta la memòria d'instruccions?
- 8) Quan es produeix el salt, quants cicles triga en executar-se la instrucció `bgt a7, zero, loop`?
- 9) Quan s'està executant la instrucció de salt, i el salt es produeix, a quina posició apunta la memòria d'instruccions?

Preguntes sobre el simulador SiMR:

- 10) Quan es produeix el salt, quants cicles triga en executar-se la instrucció `BG loop`?
- 11) Quan NO es produeix el salt, quants cicles triga en executar-se la instrucció `BG loop`?
- 12) Que bits del bus de control s'activen en el primer cicle de la instrucció `ADD R2, R3, R3`?
- 13) 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), R7`?
- 14) 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?
- 15) 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?