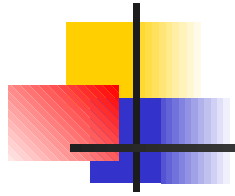


# MILLORAR EL RENDIMENT

---

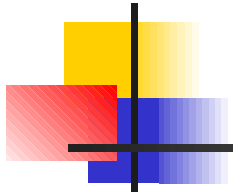
- EN UN PROCESSADOR SEQÜENCIAL
  - Simplificar més l'autòmat d'estats (Unitat de Control cablejada)
  - Millorar el microcodi (Unitat de Control Microprogramada)
  - Augmentar la velocitat del rellotge



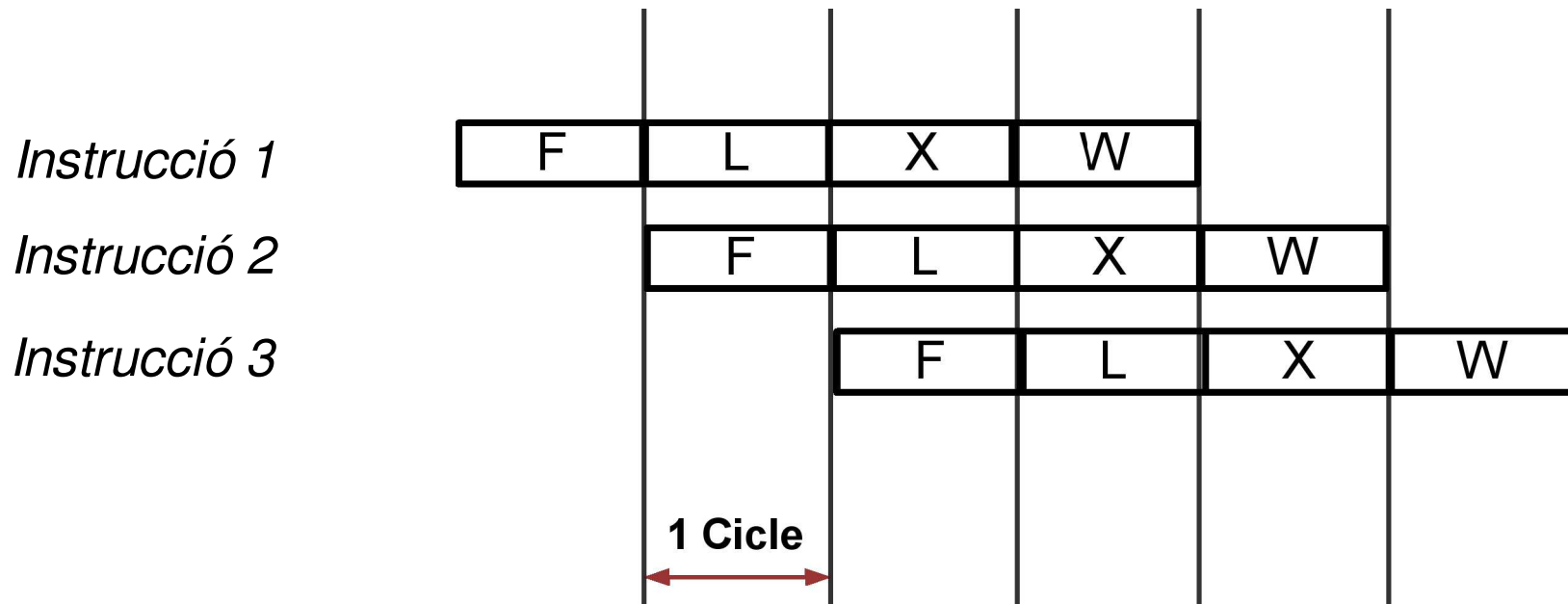
# SEGMENTACIÓ (I)

---

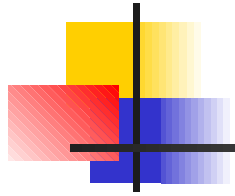
- Millorar el rendiment sense augmentar la velocitat de rellotge
- No es millora el temps d'una instrucció
- Divisió d'una instrucció màquina en etapes
  - Fetch (F), Lectura d'operands (L), Execució (E) i Escripura de resultats (W)
- La segmentació manté l'ordre d'execució de les instruccions



# SEGMENTACIÓ DE 4 ETAPES



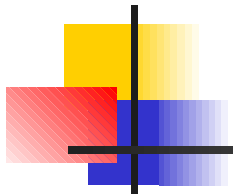
- A CADA CICLE ACABA UNA INSTRUCCIÓ



# SEGMENTACIÓ - PROBLEMES

---

- La segmentació és possible perquè en un mateix cicle, dues instruccions diferents utilitzen diferents recursos, però:
  - Què passa amb els salts?
    - Dependències de Control
  - Què passa quan una instrucció ha d'escriure una dada que la següent ha de llegir?
    - Dependències de Dades



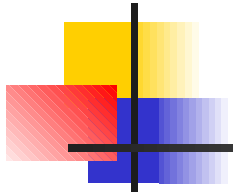
# DEPENDÈNCIES

- **DE DADES**

a	$R_i := \dots$	$a \ i \ b : \text{RAW}$
b	$:= R_i \ \text{op} \ R_j$	$b \ i \ c : \text{WAR}$
c	$R_i := \dots$	$a \ i \ c : \text{WAW} \quad (\text{No en segmentació})$

- **DE CONTROL**

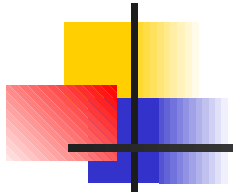




# SUPERSEGMENTACIÓ

---

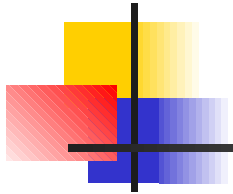
- **UN PAS MÉS:** Subdividir les etapes de la segmentació en etapes més petites:
  - Canvis estructurals: afegir més busos
- Predicció de salts
- Avantatges: més paral·lisme
- Desavantatges: més penalització
- Conflictes comuns amb la segmentació: WAR (Write After Read) i RAW (Read After Write)



# SUPERESCALARS

---

- **AFEGIR MÉS RECURSOS** (ALUs, registres, etc...) per obtenir més rendiment: vies
- Les instruccions poden **no** acabar en el mateix ordre que entren al pipeline
  - Més conflictes: WAW (Write After Write)



# SUPERESCALAR - VIES

3 VIES

