CHEATSHEET MIPS

1. DEPENDENȚE

Pentru următoarele subpuncte folosim doar dependențele RAW și de control, dar în tabel le punem pe toate: RAW (read after write)
WAW (write after write)
WAR (write after read)
de control (de la BEQ)

Cum vedem de ce tip e o dependență?

ex: instrucțiunea I_n: nume registre

```
lw      s1, s0
sw
add     s1, s2, s3
and
or
```

Fie citim dintr-un registru, fie scriem în el.
Ca să găsim dependențele, ținem cont de următoarele 3 instrucțiuni (I_{n+1}, I_{n+2}, I_{n+3}).

I. Ca să avem RAW, ne uităm la registrul în care scriem (W) la instrucțiunea I_n.
Dacă la I_{n+1}, I_{n+2}, I_{n+3} citim (R) din același registru, avem dependență.

II. Ca să avem WAW, I_{n+1}, I_{n+2}, I_{n+3} trebuie să scrie (W) în același registru în care a scris (W) I_n.

III. Ca să avem WAR, I_{n+1}, I_{n+2}, I_{n+3} trebuie să scrie (W) în registrul din care a citit (R) I_n.

LW (sau SW) S_1, $O(S_0)$ → load și store
→ sunt diferite

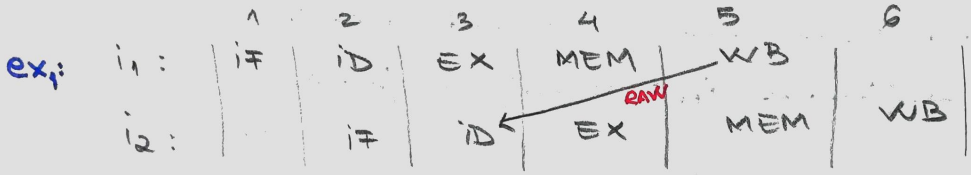
LW S_1, $O(S_0)$ vs. SW S_1, $O(S_0)$

↑ aici scriu ↑ de aici citesc — de aici citesc ↑ aici scriu

2. HAZARDURI

Fiecare instrucțiune începe la un clk după ce a început precedenta și are 5 faze:

IF fetch — ID decode — EX execute — MEM memory — WB write back



Să zicem că i_1 - i_2 are dependență RAW
⇒ trebuie să ne asigurăm că i_2 citește din registru abia după ce i_1 a scris în el

Dar în schemă vedem că WB din i_1 se face pe clk 5, iar ID din i_2 se face pe clk 3 ⇒ hazard RAW

! WB și ID pot fi pe același clk pt. că WB se face pe primul front, iar ID pe al doilea

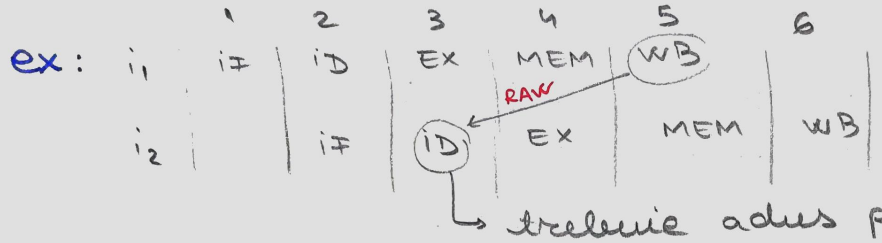
ex2: Să zicem că i_1 - i_2 are dependență de control
⇒ trebuie să ne asigurăm că I_2 face IF după ce i_1 face MEM (IF trebuie să fie cel puțin cu un clk mai în dreapta)

Hazardurile RAW se numără normal, dar cele de control în funcție de câte BEQ avem, nu de câte săgețele cu control avem pe schemă.

METODE DE A SCĂPA DE HAZARDURI

① Păcănele - încercăm să rearanjăm codul cât să avem mai puține hazarduri

② NOP = băgăm delay-uri (instrucțiuni goale) ca să ajungă fazele instrucțiunilor pe ce clk urem

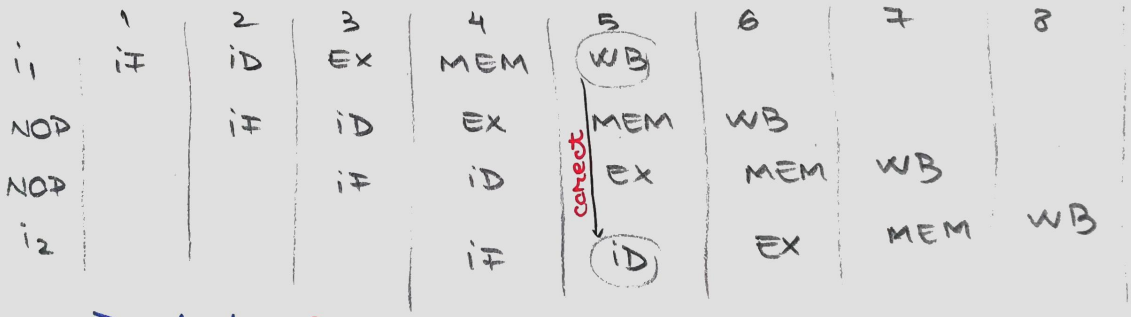


↳ trebuie adus pe clk 5

NOP: IF ID EX MEM WB

=> trebuie mutat cu 2 clk-uri

=> băgăm 2 NOP-uri



=> am scăpat de RAW

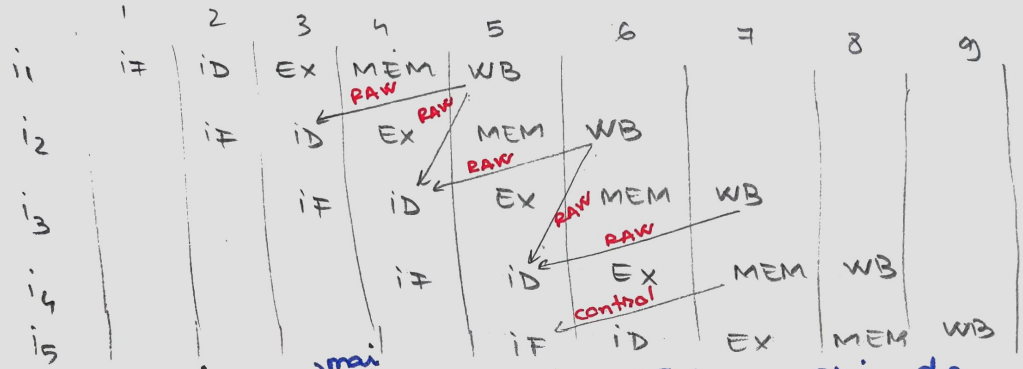
! Când e un cod mai complex, e posibil ca rezolvând unele hazarduri, să se rezolve și altele automat

! La fel și cu hazardurile de control, doar că IF trebuie să ajungă la minim un clk după MEM

! Tot codul ia mai mult timp de procesare acum, poate putem să îmbunătățim: ① + ② sau ③

③ STALL = întârziem fazele unor instrucțiuni pentru a fi făcute când vrem noi

ex:



- Întârziem ID (îl mai scriem de atâtea ori de câte clk-uri trebuie shiftat)

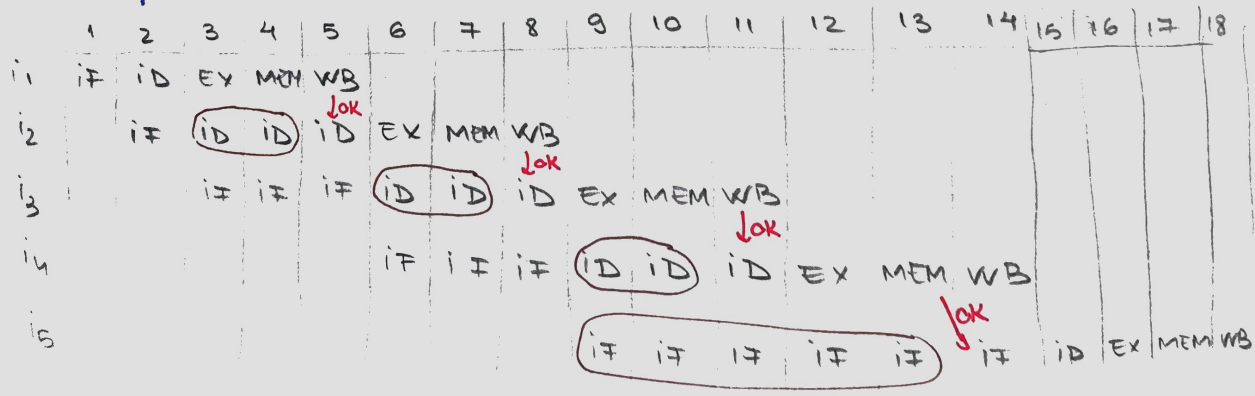
ID de la i_3 e la clk 4 } => mai scriem 2 ID-uri
WB de la i_2 e la clk 6

- Pt. instrucțiunea de dedesubt vom scrie IF → începând cu primul clk cu ID de deasupra și până la ultimul ID de deasupra inclusiv (RAW)

SAU

până la WB de deasupra (control)

Deci pt. ex. de sus:



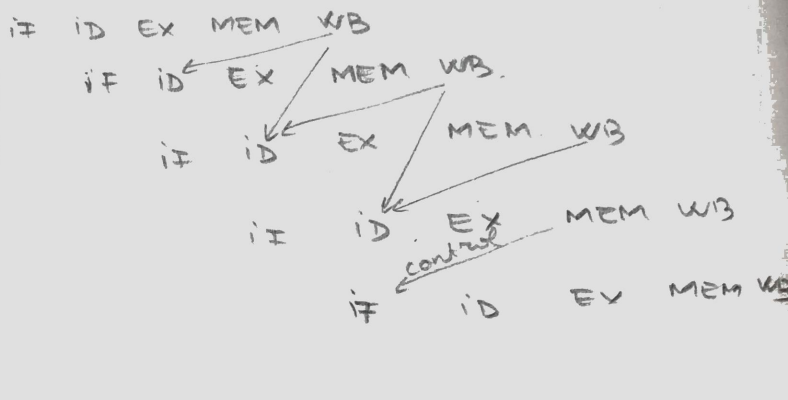
④ Forwarding

Facem stall astfel încât pentru instrucțiunile

- LW și SW : dif. de 1 clk între MEM și EX
- celelalte : dif. de 1 clk între EX și EX

Ex :

```
i1   lw  s1, 0(s2)
i2   lw  s3, 0(s1)
i3   add s1, s1, s3
i4   beq s2, s3, ET
i5   OR  s2, s2, s3
ET : nimic
```



	1	2	3	4	5	6	7	8	9	10	11	12
i_1	IF	ID	EX	MEM	WB							
i_2		IF	ID	ID	EX	MEM	WB					
i_3			IF	IF	ID	ID	EX	MEM	WB			
i_4					IF	IF	ID	ID	EX	MEM	WB	
i_5							IF	IF	ID	EX	MEM	WB