

Architecture des microprocesseurs

RISC-V(RV32I)

control de flot : sauts et branchements

Michel Agoyan
Marc Lacruche
Simon Pontie
Olivier Potin
Come Allart
Raphaël Comps

: michel.agoyan@st.com
: marc.lacruche@st.com
: simon.pontie@cea.fr
: olivier.potin@emse.fr
: come.allart@emse.fr
: rcomps@emse.fr

Architecture µP: RV32I saut et branchements

* RISC-V (RV32I) format des instructions

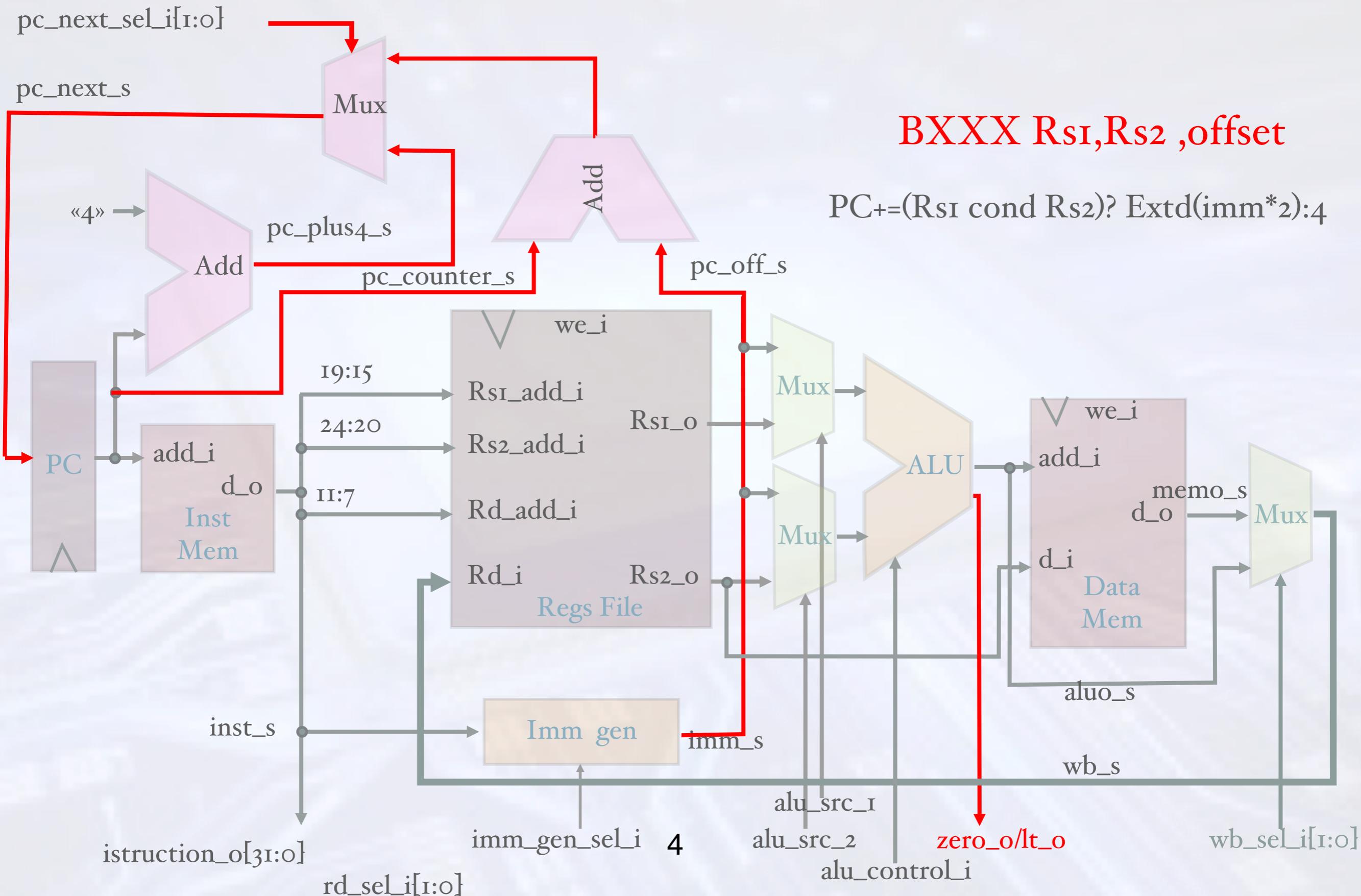
| | 31 ... 25 | 24 ... 20 | 19 ... 15 | 14 ... 12 | 11 ... 7 | 6 ... 0 |
|----|----------------------------------|-----------|-----------|-----------|-------------------|---------|
| R | func7 | rs2 | rs1 | funct3 | rd | 0110011 |
| I | imm[11:0] | | rs1 | funct3 | rd | 0010011 |
| S | Imm[11:5] | rs2 | rs1 | funct3 | imm[4:0] | 0100011 |
| SB | imm[12] imm[10:5] | rs2 | rs1 | funct3 | imm[4:1] imm[1] | 1100111 |
| U | Imm[31:12] | | | | rd | 0110111 |
| UJ | imm[20] imm[10:1] imm[19:12] | | | | rd | 1101111 |

RV32I

Sauts et Branchements

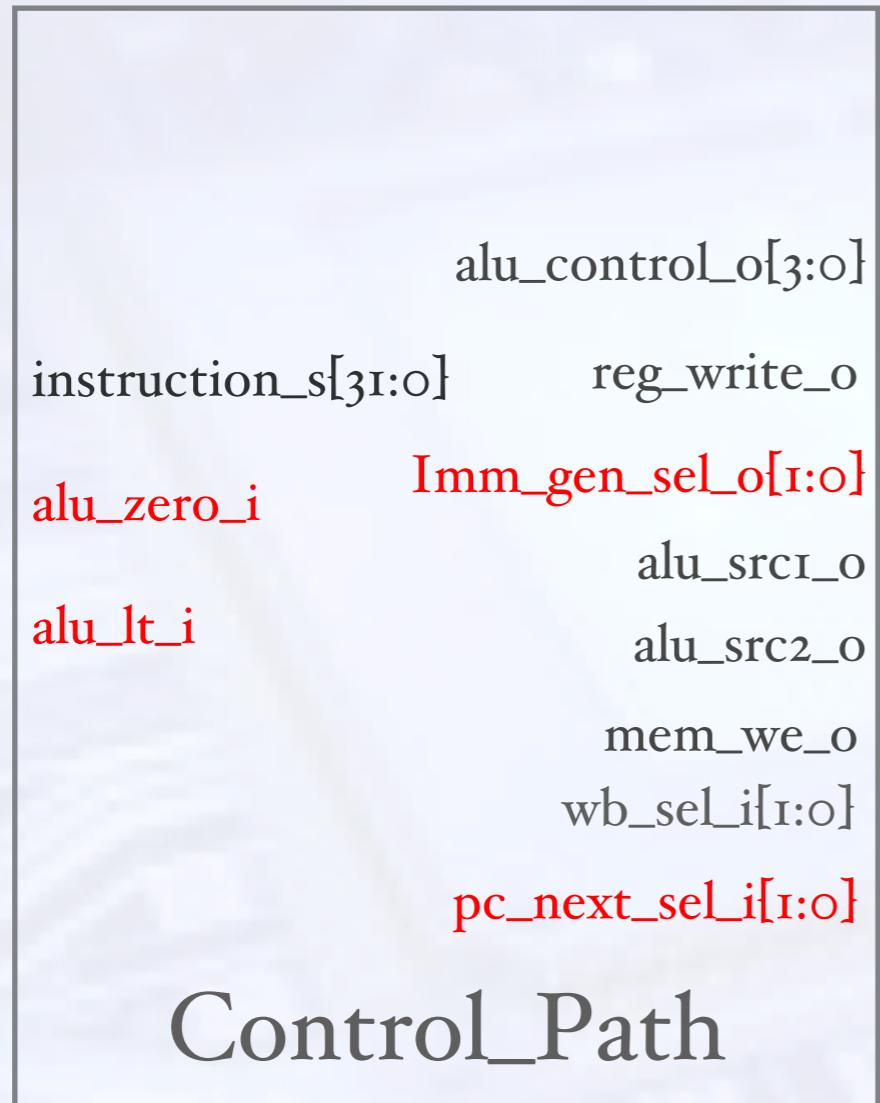
Architecture monocycle

construction du data-path pour lebranchement



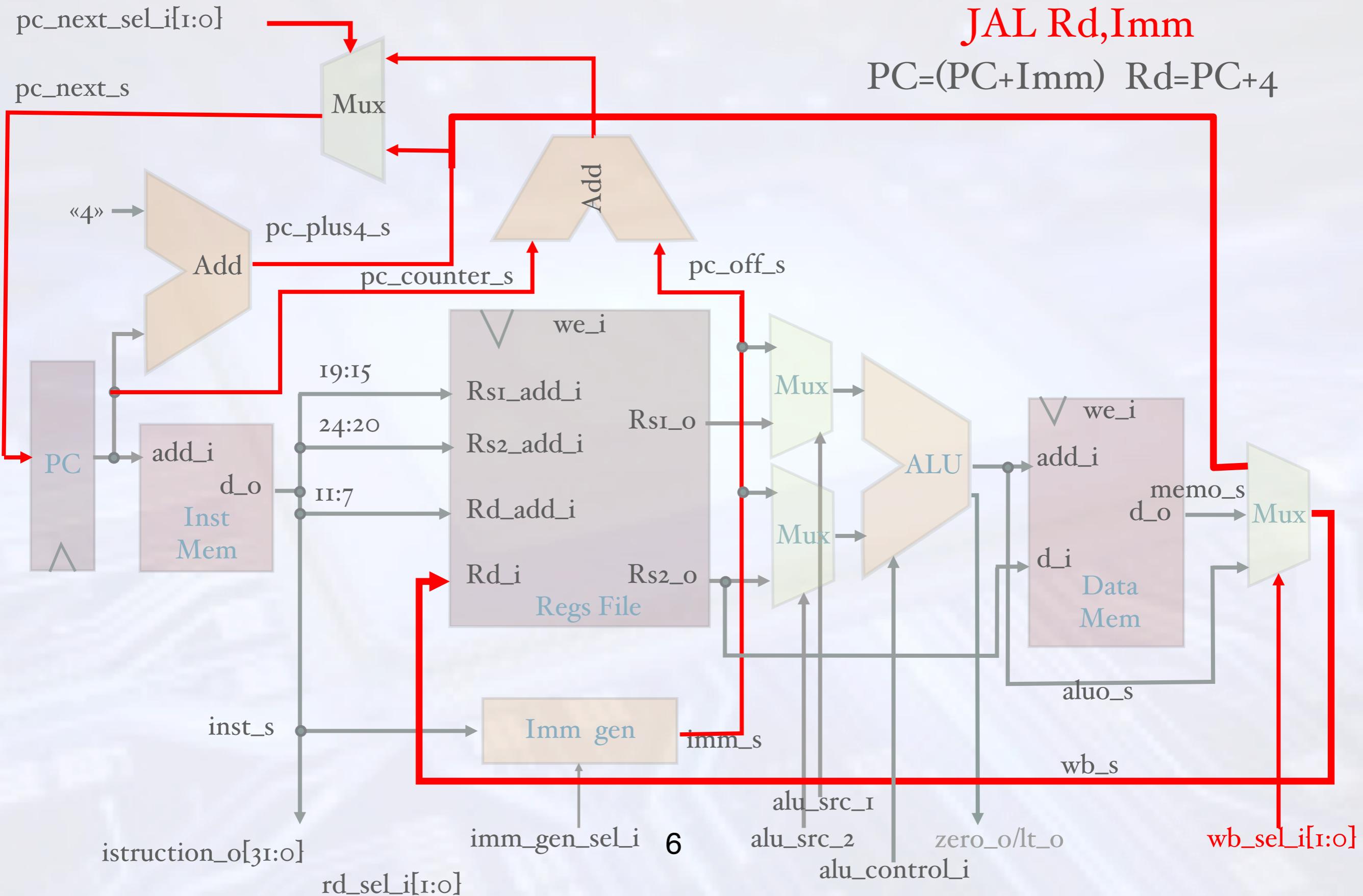
Logique de contrôle du data-path Pour les instructions de branchement BXXXX

Rs1,Rs2,offset de format SB



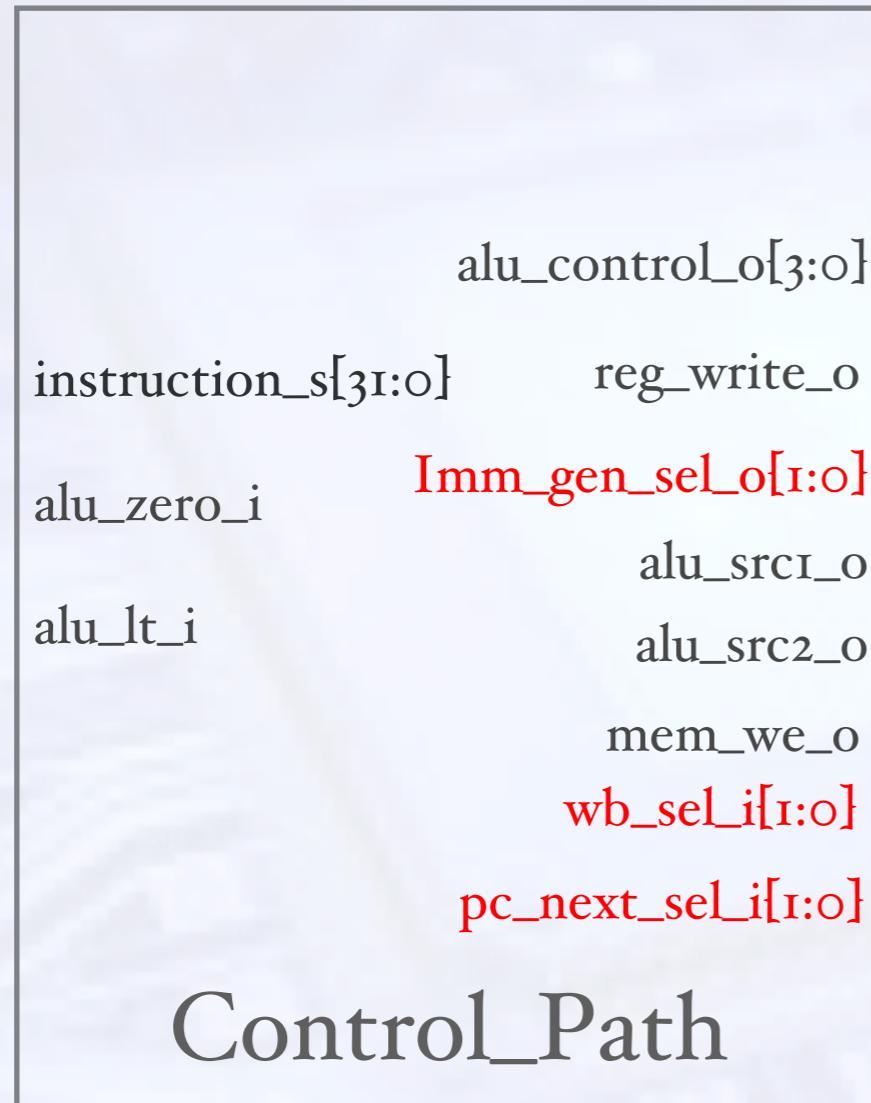
- * On ajoute à la logique du « control path » :
- * Le décodage d'instructions de type SB qui va permettre de :
 - * générer le signal de pilotage « `pc_next_sel_i` » en fonction des bits de statut de l'ALU
 - * D'appliquer le traitement adéquat sur la valeur immédiate
 - * De choisir la bonne opération pour l'ALU (SUB..)

construction du data-path pour le saut



logique de contrôle du data-path

Pour les instructions de saut JAL Rd,Imm de format UJ



- * On ajoute à la logique du « control path » :
- * Le décodage d'instructions de type UJ qui va permettre de :
 - * générer le signal de pilotage « pc_next_sel_i»
 - * D'appliquer le traitement adéquat sur la valeur immédiate
 - * De sectionner l'entrée du multiplexeur pour le WB

RV32I

Sauts et Branchements

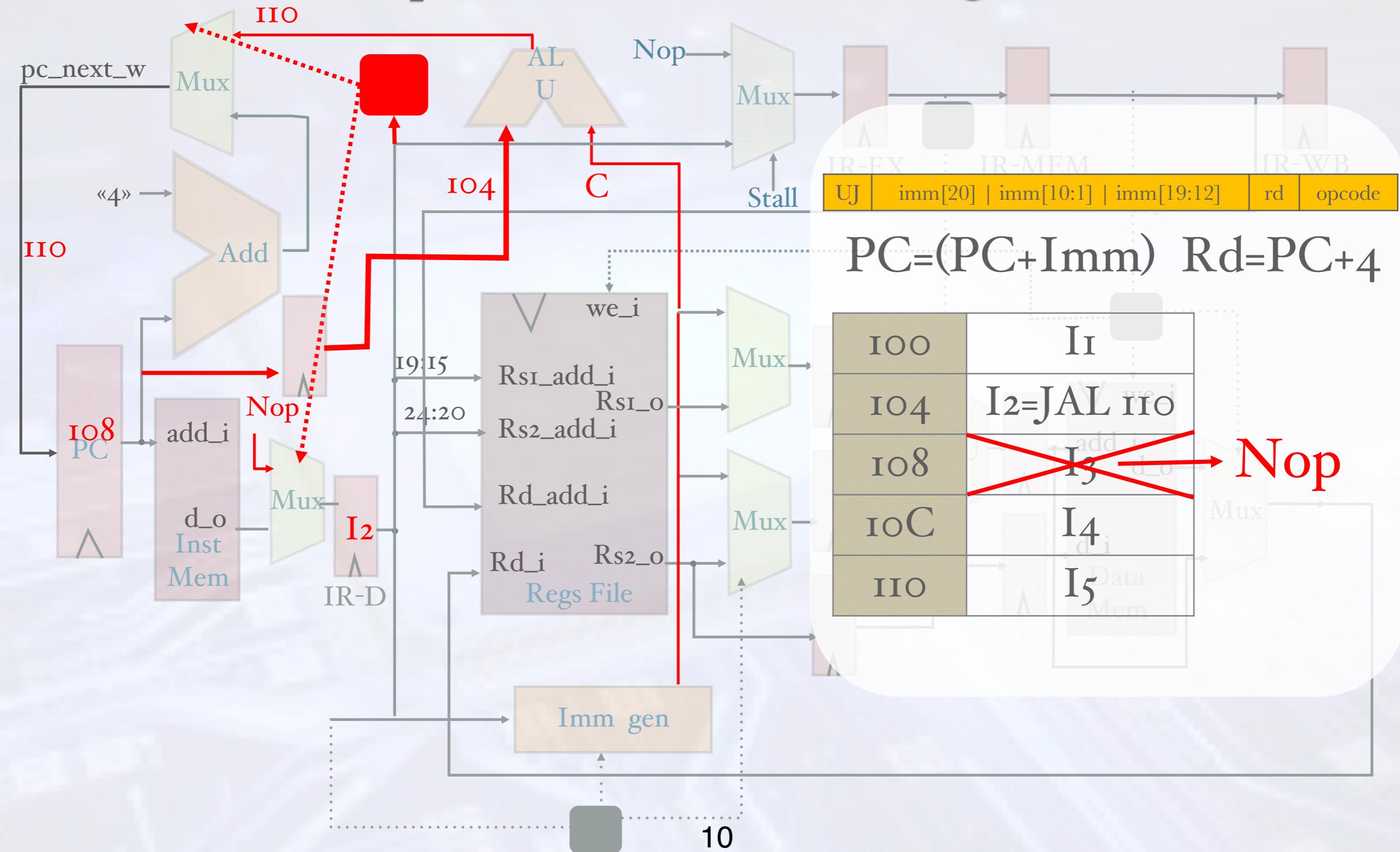
Architecture pipeline

pipeline : dépendance de contrôle

- * En cas de rupture du flot d'exécution par une instruction de saut ou de branchement comment est calculée l'adresse de destination (architecture de type pipeline RV32I) ?
- * JAL saut à PC + immédiate
- * JALR saut indirect à Rsi+ immédiate
- * Bxxx si (Rsi cond Rs2) branchement à PC+Imm

| instruction | Étage pipeline pour le calcul de décision | Etage pipeline de calcul d'adresse de destination |
|-------------|---|---|
| JAL | Decode | Decode |
| JALR | Decode | Execution |
| Bxxx | Execution | Decode |

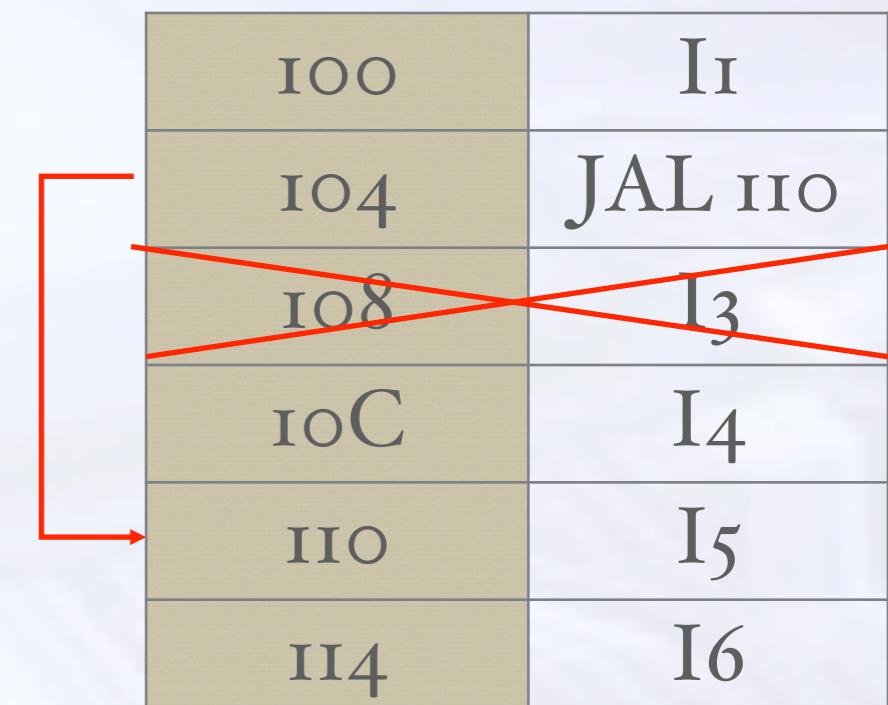
Dépendance de contrôle JAL



pipeline dépendance de contrôle : JAL

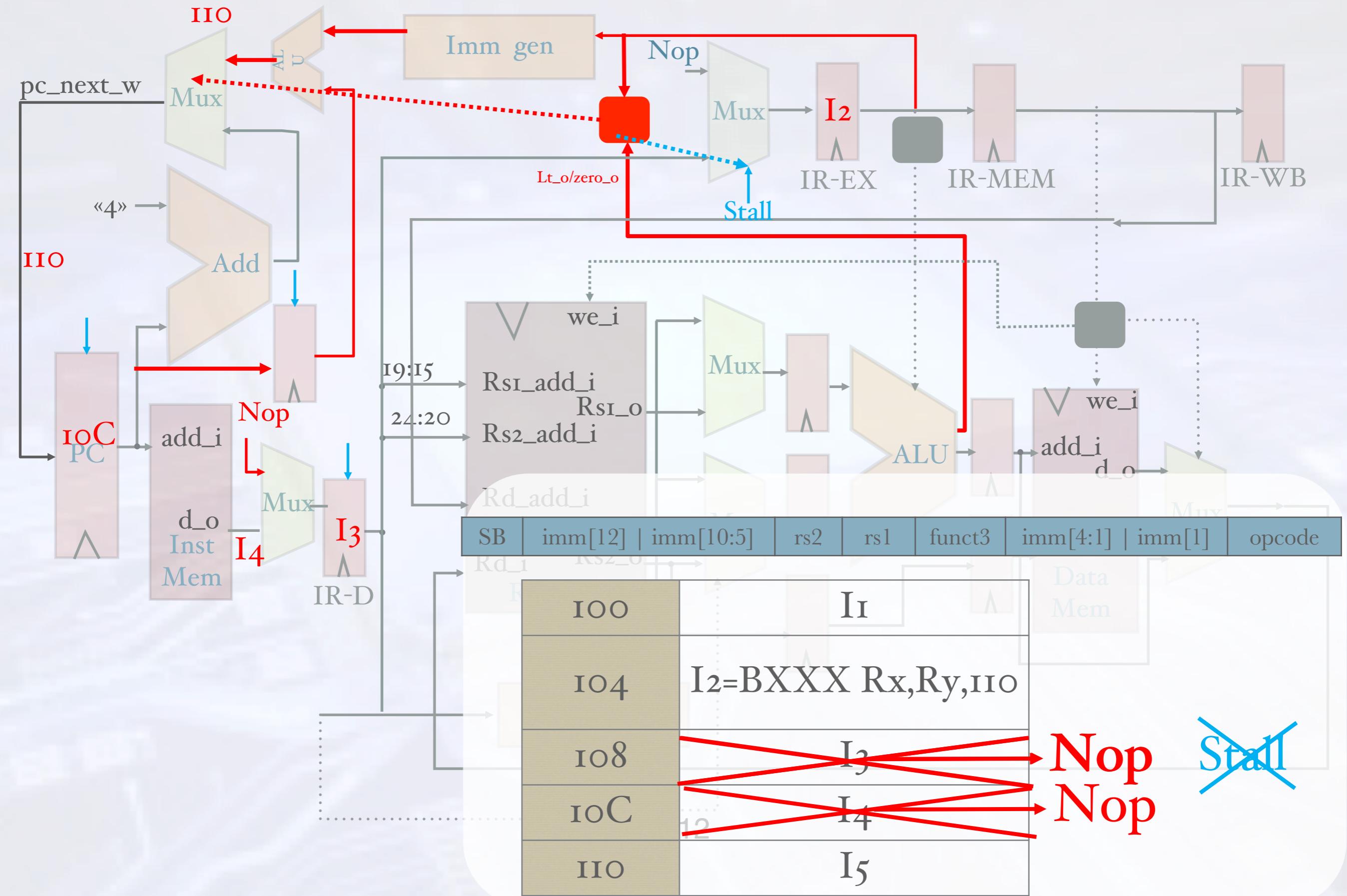
| | IF | ID | EX | MEM | WB | | | |
|---------------------|-----------------|-----------------|-----------------|------------------|------------------|-----------------|------------------|------------------|
| | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 |
| I ₁ | IF ₁ | ID ₁ | EX ₁ | MEM ₁ | WB ₁ | | | |
| I ₂ =JAL | | IF ₂ | ID ₂ | EX ₂ | MEM ₂ | WB ₂ | | |
| I ₃ | | | IF ₃ | NOP | NOP | NOP | NOP | |
| I ₅ | | | | IF ₅ | ID ₅ | EX ₅ | MEM ₅ | WB ₅ |
| I ₆ | | | | | IF ₆ | ID ₆ | EX ₆ | MEM ₆ |

| | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 |
|-----|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| IF | I ₁ | I ₂ | I ₃ | I ₄ | I ₆ | I ₇ | I ₈ | I ₉ |
| ID | | I ₁ | I ₂ | NOP | I ₄ | I ₆ | I ₇ | I ₈ |
| EX | | | I ₁ | I ₂ | NOP | I ₄ | I ₆ | I ₇ |
| MEM | | | | I ₁ | I ₂ | NOP | I ₄ | I ₆ |
| WB | | | | | I ₁ | I ₂ | NOP | I ₄ |



Impact sur CPI ↗
 Pas d'impact sur le signal **stall**

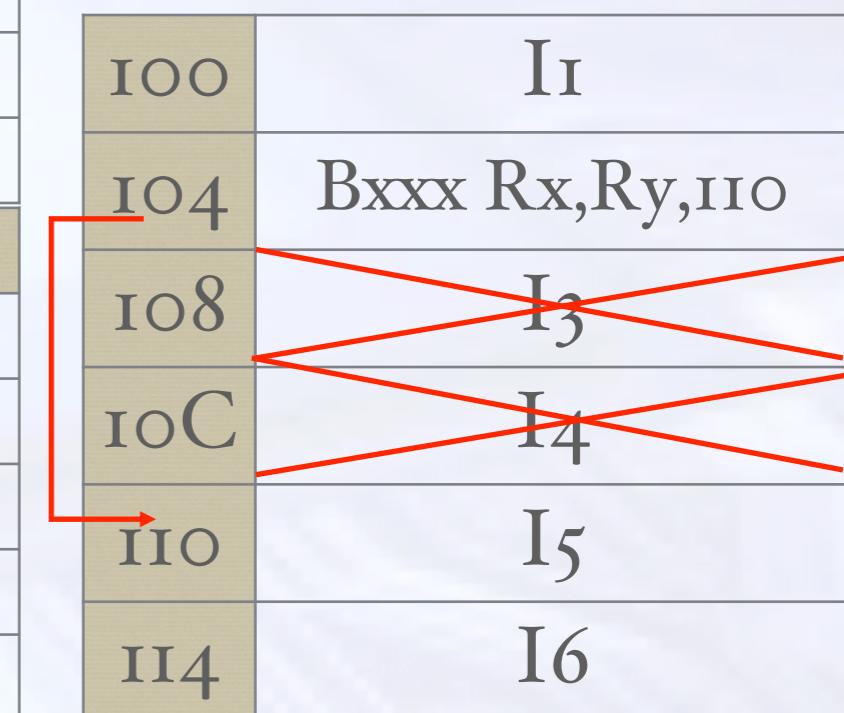
Dépendance de contrôle BXXX



pipeline dépendance de contrôle : BXXX

| | IF | ID | EX | MEM | WB |
|--|----|----|----|-----|----|
|--|----|----|----|-----|----|

| | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 |
|----------------------|-----------------|-----------------|-----------------|------------------|------------------|-----------------|-----------------|------------------|
| I ₁ | IF ₁ | ID ₁ | EX ₁ | MEM ₁ | WB ₁ | | | |
| I ₂ =BXXX | | IF ₂ | ID ₂ | EX ₂ | MEM ₂ | WB ₂ | | |
| I ₃ | | | IF ₃ | ID ₃ | NOP | NOP | NOP | |
| I ₄ | | | | IF ₄ | NOP | NOP | NOP | NOP |
| I ₅ | | | | | IF ₅ | ID ₅ | EX ₅ | MEM ₅ |
| | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 |
| IF | I ₁ | I ₂ | I ₃ | I ₄ | I ₅ | I ₆ | I ₇ | I ₈ |
| ID | | I ₁ | I ₂ | I ₃ | NOP | I ₅ | I ₆ | I ₇ |
| EX | | | I ₁ | I ₂ | NOP | NOP | I ₅ | I ₆ |
| MEM | | | | I ₁ | I ₂ | NOP | NOP | I ₅ |
| WB | | | | | I ₁ | I ₂ | NOP | NOP |



Impact sur CPI ↗↗

Impact sur le signal Stall:

Par exemple pour BEQ

stall'=stall*((opcode==BEQ)*zero_o)

RV32I

Sauts et Branchements

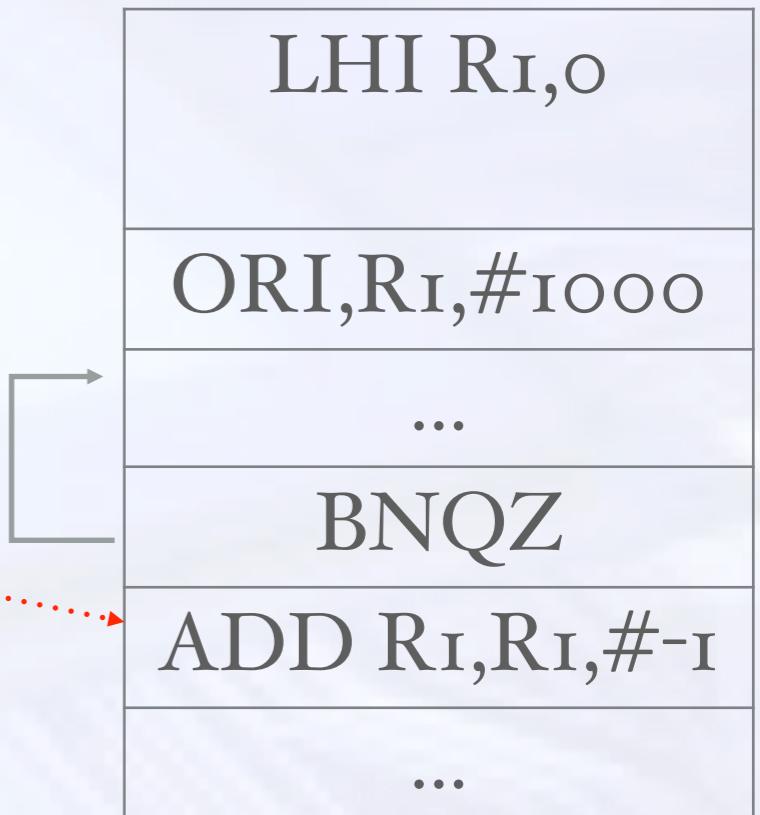
Architecture pipeline

Optimisation

- Prédiction de branchement (Branch Prediction)
- Table d'adresses de destination (Branch Target Buffer)

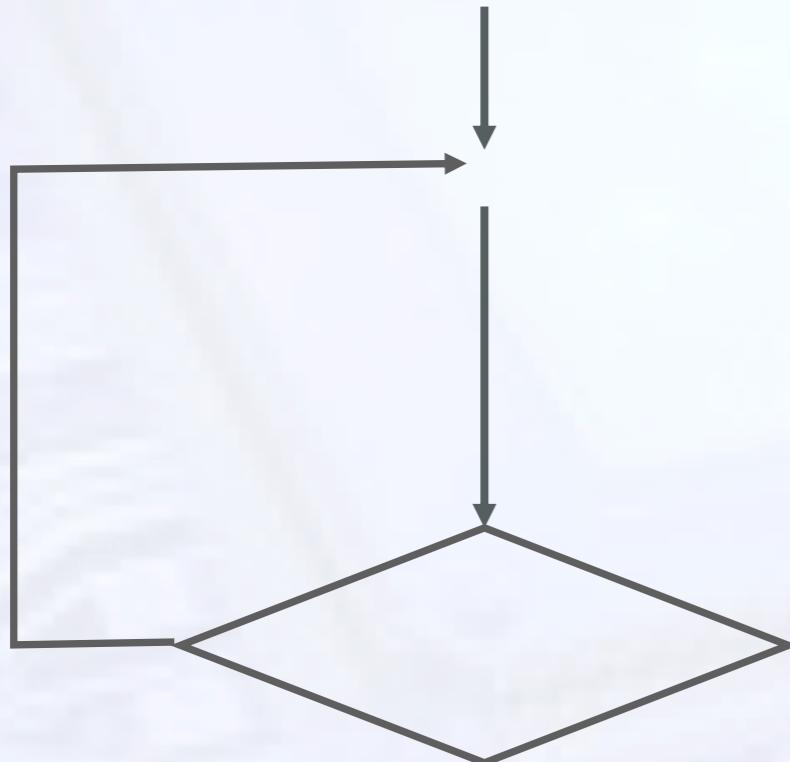
pipeline dépendance de contrôle : optimisation MIPS I

- * Les branchements avec test sont utilisés pour les boucles \Rightarrow Impact important sur CPI
- * Idée \Rightarrow Ne pas détruire l'instruction qui suit l'instruction de saut ou de branchement = « délai slot »
- * Pas forcément une bonne idée du point de vue abstraction : la microarchitecture a un impact sur l'ISA et sur les couches supérieures :
 - * l'évolution de la microarchitecture (nombre d'étages par exemple du pipeline) devient difficile

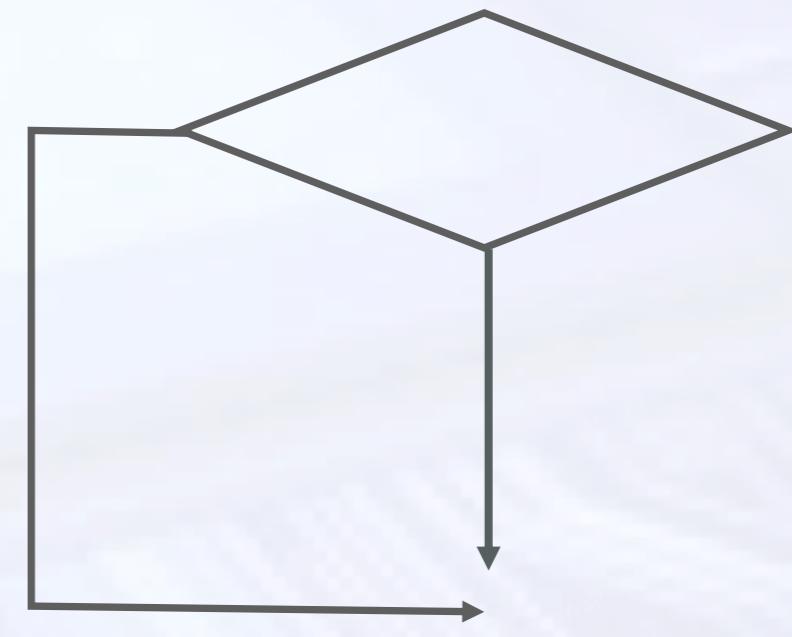


Prédiction de branchement statique

Probabilité de branchement :~ 60% (Benchmark)



Boucle c:
Branchement Arrière
Probabilité : ~90%

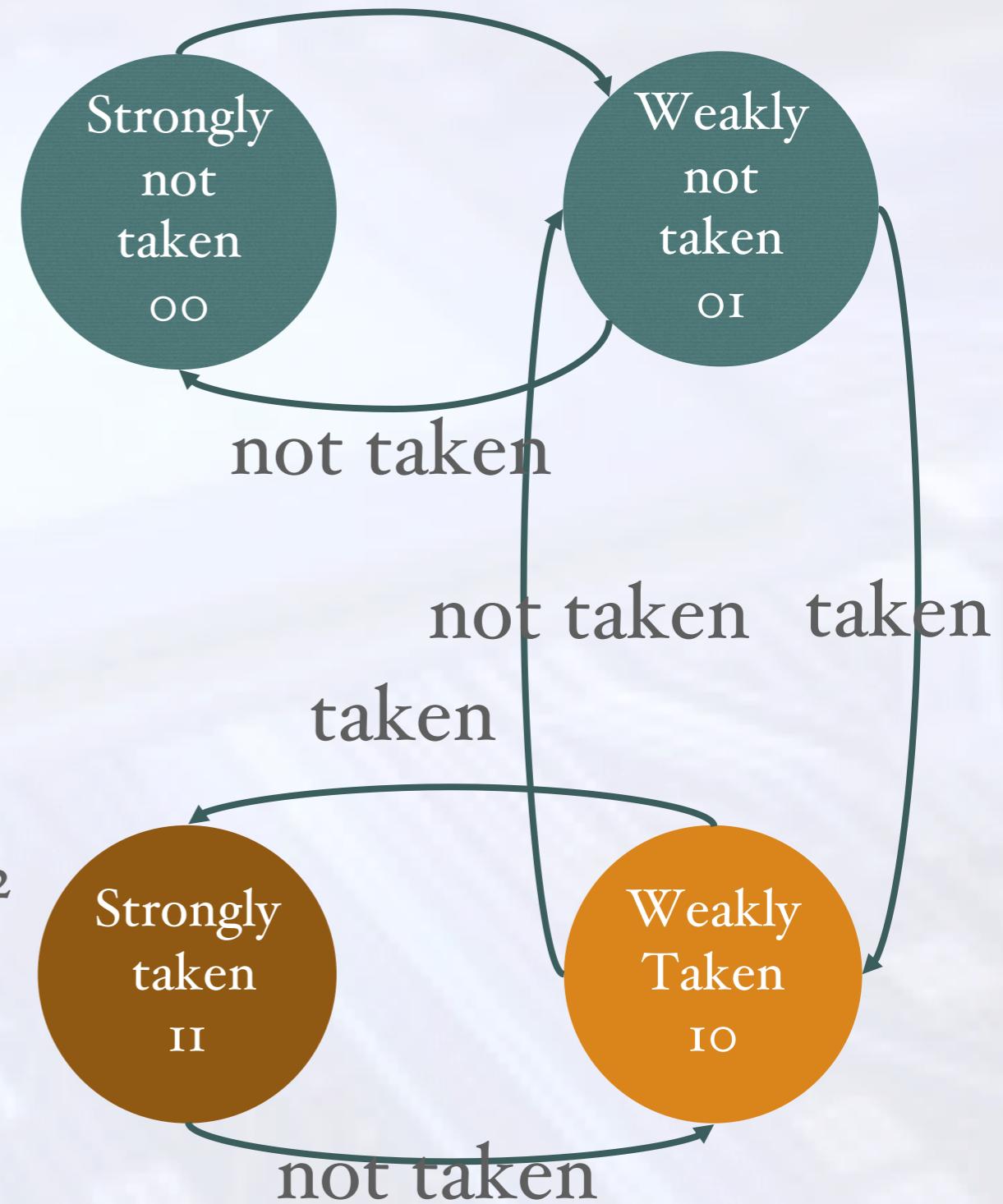


Switch case c:
Branchement Avant
Probabilité : ~50%

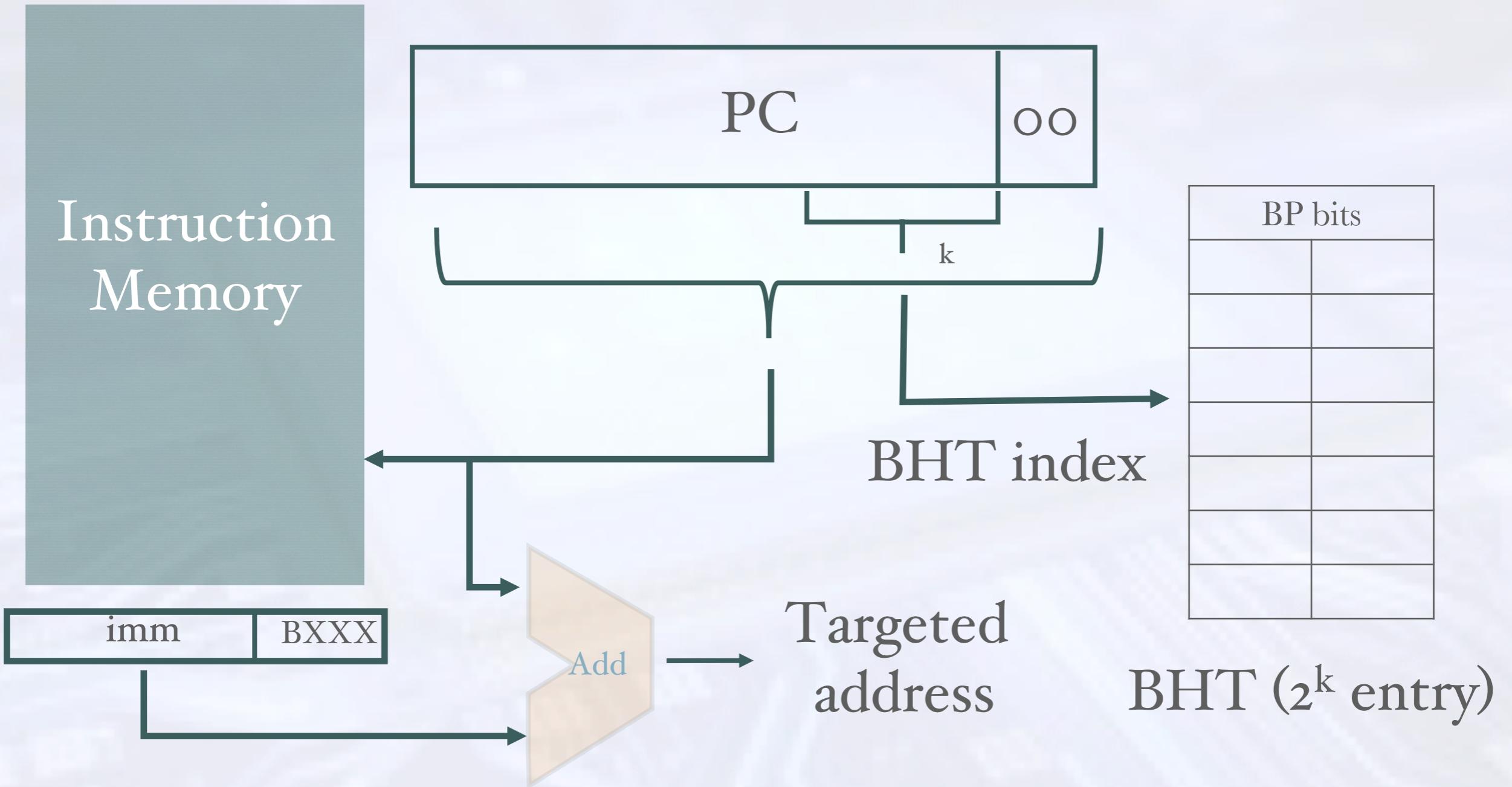
Prédiction de branchement

Dynamique: compteur _{taken}

- * Basée sur :
- * Corrélation temporelle
 - * Historique d'exécution
- * Corrélation spatiale
 - * Chemin d'exécution préférentiel
- * On associe 2 BP bits/instruction
- * On change le type de prédiction après 2 erreurs consécutives de prédiction



Prédiction dynamique de branchement: “Branch History Table”



- Une table à 4096 entrées assure ~ 90 % de prédictions justes

Prédiction de branchement dynamique: “Branch Target Buffer”

- * On associe à la BTH (“Branch Target History”) un BTB (“Branch Target Buffer”)
 - * Dans l’étage de “Fetch” on charge l’instruction issue du mécanisme de prédiction
 - * Si la prédiction est fausse, on supprime l’instruction chargée
 - * Dans tous les cas on met à jour BHT et BTB

