Architecture des processeurs RISC-V (RV32I)

Architecture multicycle, Pipeline

Michel Agoyan (michel.agoyan@thalesgroup.com),
Clément Fanjas (clement.fanjas@cea.fr),
Théophile Gousselot (theophile.gousselot@emse.fr),
Marc Lacruche (marc.lacruche@st.com),
Louis Noyez (louis.noyez@emse.fr),
Simon Pontié (simon.pontier@cea.fr),
Olivier Potin (olivier.potin@emse.fr),
Jean-Baptiste Rigaud (rigaud@emse.fr).

3 janvier 2023

Notes

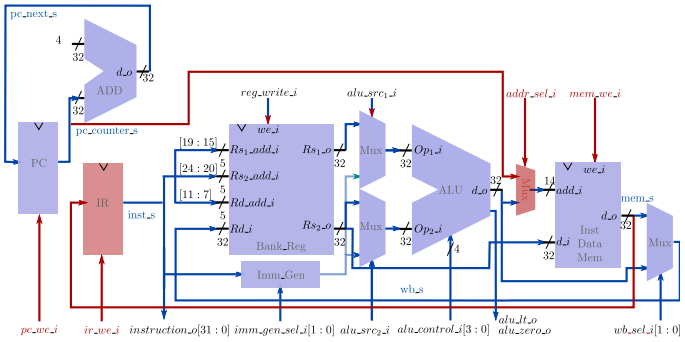
Outline

Notes

Plan

Notes

Data path

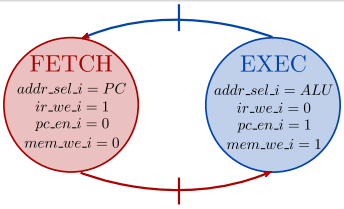


Notes

Control path

2 états

- Fetch
- Execute



Notes

Performances

Pour l'état Fetch

- Le chemin critique vaut : t_{ifetch}

Pour l'état Execute

- Le chemin critique vaut : $t_{opfetch} + t_{ALU} + t_{MEM}$

Donc :

- $T_{bicycle} < T_{monocycle}$
- si $t_{ifetch} = T_{MEM} >> t_{opfetch} + t_{ALU}$ alors $T_{monocycle} = 2T_{bicycle}$
- Pas de différence de performance

Notes

Plan

Notes

Architecture mono-cycle vs règles d'optimisation

- Réduction des chemins critiques
  - Les chemins se superposent !
- Optimiser les opérations les plus fréquentes
  - Le traitement le plus long impose le temps de cycle !
- Équilibrer l'utilisation du matériel
  - Si une ressource doit être utilisée plusieurs fois elle doit être dupliquée !

Idée

- Plusieurs cycles/Instruction = micro-architecture multi-cycle

⇒ Découpler le temps de cycle du temps de traitement de l'instruction

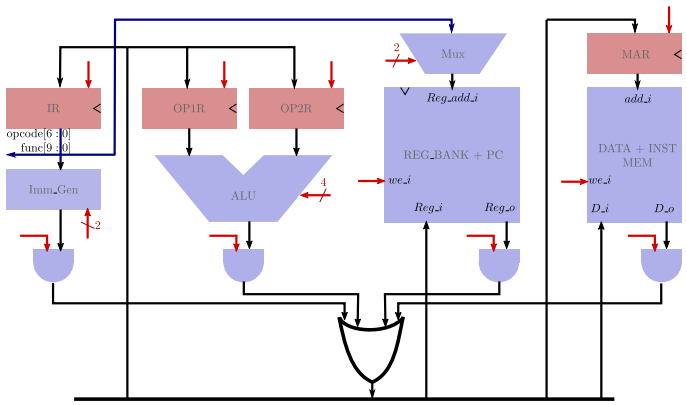
Notes

Architecture multi-cycle vs règles d'optimisation

- Réduction des chemins critiques
  - Le chemin critique peut être optimisé pour chaque instruction
- Optimiser les opérations les plus fréquentes
  - Les instructions deviennent indépendantes, d'un point de vue temporel et on peut donc optimiser les opérations les plus fréquentes
- Équilibrer l'utilisation du matériel
  - On peut mieux factoriser le matériel entre les instructions ou pour une instruction

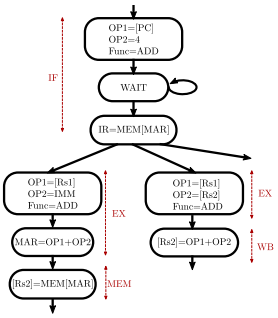
Notes

Le data-path : 18 signaux de contrôle



Notes

Le control-path

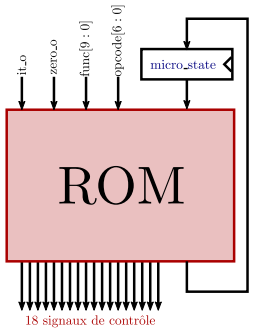


On distingue 5 opérations élémentaires :

- IF : récupération de l'instruction (fetch) et décodage d'instruction (decode)
- ID : récupération des opérandes (fetch operands)
- EX : exécution
- MEM : accès mémoire
- WB : l'écriture du résultat dans un registre (write back)

Notes

Le control-path micro-programmé



Les instructions sont des micro-séquences :

- F0 : Op1 = PC ; Op2 = 4
- F1 : PC = Op1 + Op2
- F2 : MAR = PC
- F3 : IR = MEM[MAR]

Pour chaque micro-état on définit :

- les 18 signaux de contrôle
- le micro-état suivant

Taille de la ROM :

- $2^{(sizeof(opcode|func)+2+sizeof(state))} \times (18 + sizeof(state))$!

Performances :

- CPI différent pour chaque instruction
- Temps de cycle optimal

Notes

Plan

Notes

Micro-architecture pipeline

La microarchitecture multi cycle permet une bonne factorisation des ressources matérielles mais ne permet pas le parallélisme des traitements.

- Compromis entre la factorisation des ressources matérielles et la duplication avec parallélisme

Est-ce que l'architecture mémoire Harvard est une si mauvaise idée ?

- Elle nous a permis d'accéder à la fois à l'instruction et à la donnée dans l'architecture monocycle (afin de résoudre la dépendance structurelle)
- Compromis entre une architecture Von Neumann et Harvard : Hiérarchie mémoire (prochain cours)

Comment paralléliser des traitements ? :

- "le travail à la chaîne !" ⇒ le pipeline

Notes

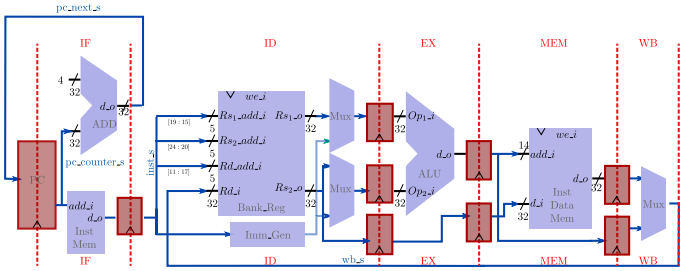
Pipeline idéal

Nous reprenons nos 5 traitements de base de l'architecture multi-cycle que nous plaçons sur une chaîne de traitement⇒ le pipeline.
Les hypothèses d'une chaîne de traitement pour garantir un usage optimal des ressources de traitement sont les suivantes :

- Toutes les instructions passent par les mêmes étapes (Stages)
- Pas de ressources partagées entre deux étapes
- Les délais de traitement de chaque étape sont égaux
- L'entrée d'une instruction dans le pipeline est cadencée par la durée de traitement

Notes

Pipeline 5 étages : data path



Notes

Pipeline 5 étages : illustration

IF	ID	EX	MEM	WB

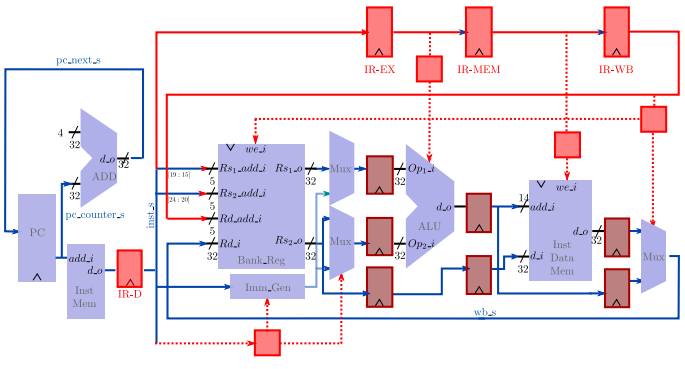
	c1	c2	c3	c4	c5	c6	c7	c8
I1	IF1	ID1	EX1	MEM1	WB1			
I2		IF2	ID2	EX2	MEM2	WB2		
I3			IF3	ID3	EX3	MEM3	WB3	
I4				IF4	ID4	EX4	MEM4	WB4
I5					IF5	ID5	EX5	MEM5

	c1	c2	c3	c4	c5	c6	c7	c8
IF	I1	I2	I3	I4	I5	I6	I7	I8
ID		I1	I2	I3	I4	I5	I6	I7
EX			I1	I2	I3	I4	I5	I6
MEM				I1	I2	I3	I4	I5
WB					I1	I2	I3	I4

latence = 5 cycles — 1 instruction / cycle

Notes

Pipeline 5 étages (contrôle du data path)



Notes

Plan

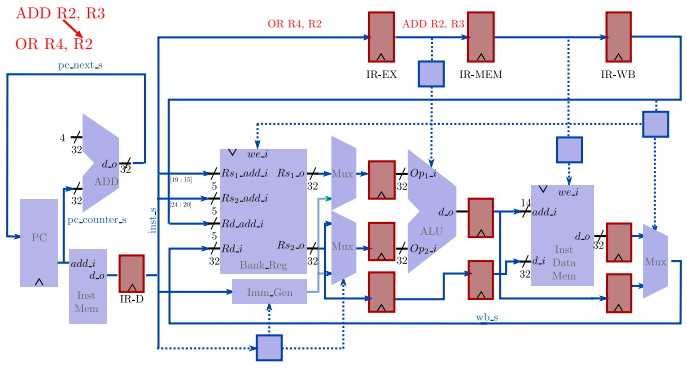
Notes

Les dépendances ("Hazard")

- Nous avons vu précédemment qu'une concurrence sur une même ressource matérielle peut entraîner une dépendance structurelle («structural hazard») entre deux traitements
  - Dans le pipeline deux instructions peuvent interagir l'une sur l'autre à cause d'une dépendance structurelle (accès mémoire)
- Une instruction peut aussi dépendre de ce qu'a produit l'instruction précédente
  - Dépendance de données («data hazard»)
  - Dépendance de contrôle («control hazard») : calcul du PC

Notes

Les dépendances de données



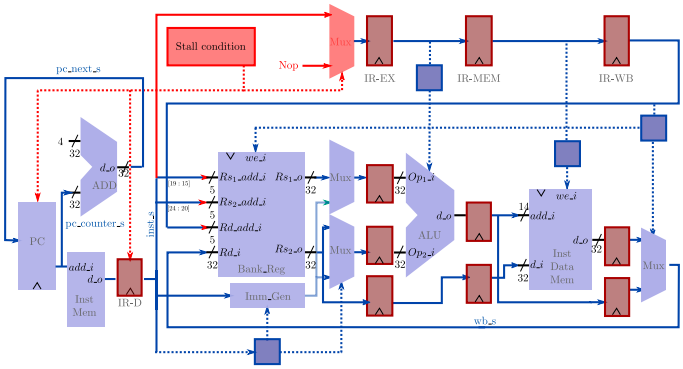
Notes

Résolution des dépendances de données

- Réordonner les instructions pour supprimer les dépendances et le cas échéant insérer des « NOP » (*no operation*)
  - Stratégie du MIPS I (*Microprocessor without Interlocked Pipeline Stage*)
- Attendre le résultat en arrêtant les étages qui précèdent : « Interlocks »
- Trouver des raccourcis pour mettre les données à jour le plus tôt possible : « Bypass »

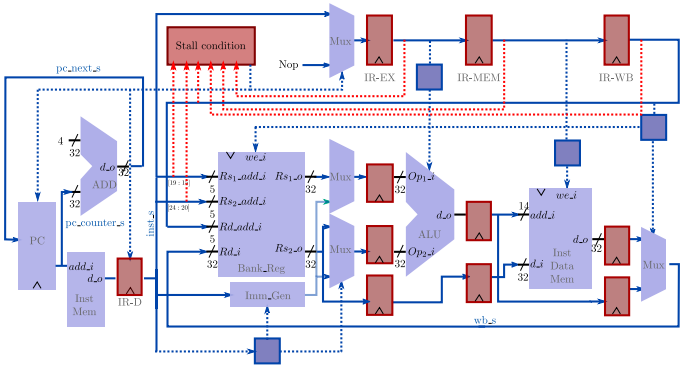
Notes

Interlocks



Notes

Interlocks : logique de contrôle



$stall = (Rs1\_add == Rd\_add) || (Rs2\_add == Rd\_add)$

Notes

Génération du signal de contrôle stall

- $stall = (Rs1\_add == Rd\_add)||(Rs2\_add == Rd\_add)$
- Affiner les conditions de manière à ne figer le pipeline que si nécessaire

Format Inst		Sources	Destination	
ALU R	Rd = Rs1 func Rs2	Rs1,Rs2	Rd	R0
ALU I	Rd = Rs1 func Imm	Rs1	Rd	R0
S	MEM[Rs1+Imm]=Rs2	Rs1,Rs2		
J	PC=Pc+Imm		Rd	R0

Notes

Interlock : illustration

IF	ID	EX	MEM	WB

		c1	c2	c3	c4	c5	c6	c7	c8
I1	ADD R2, R3	IF1	ID1	EX1	MEM1	WB1			
I2	OR R4, R2		IF2	ID2	ID2	ID2	ID2	EX2	MEM2
I3				IF3	IF3	IF3	IF3	ID3	EX3
I4								IF4	ID4
I5									

Notes

Bypass : principe

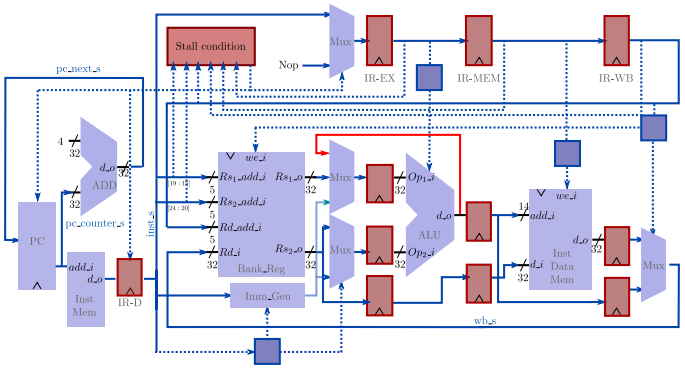
IF	ID	EX	MEM	WB

		c1	c2	c3	c4	c5	c6	c7	c8
I1	ADD R2, R3	IF1	ID1	EX1	MEM1	WB1			
I2	OR R4,R2		IF2	ID2	EX2	MEM2	WB2		
I3				IF3	ID3	EX3	MEM3	WB3	
I4					IF4	ID4	EX4	MEM4	WB4
I5						IF5	ID5	EX5	MEM5

- En fait le résultat de l'exécution de I1 est disponible à l'étage EX et peut être fourni à temps pour l'exécution de I2
- Il faut pour cela créer un bypass entre la sortie de l'ALU et son entrée

Notes

Bypass : circuit



Notes

Plan

Notes

Conclusion

- CPI >1
- Bypass complet augmente la complexité et a un impact sur le chemin critique et le temps de cycle.
- Dépendance de donnée entre un LW et l'instruction suivante ne peut pas être résolu par bypass
- Dépendances de contrôle dues aux branchements conditionnels non résolues par bypass (cf. séance 4)
- Hypothèse forte :
  - Structure mémoire Harvard avec un temps d'accès = 1 cycle
  - Séance 5 : hiérarchie mémoire

Notes