ΗΥ425 - Αρχιτεκτονική Υπολογιστών - Μ3

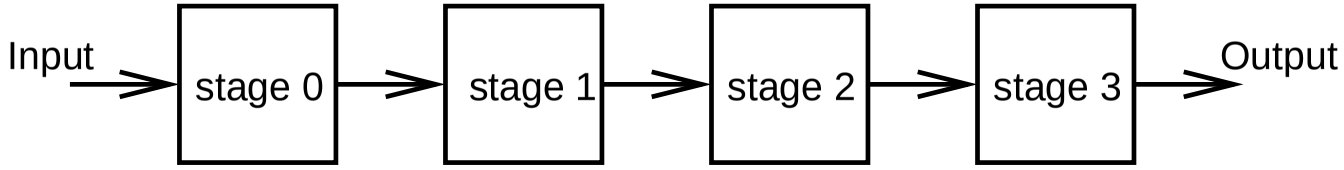
Χ. Σωτηρίου

13 Οκτωβρίου 2001

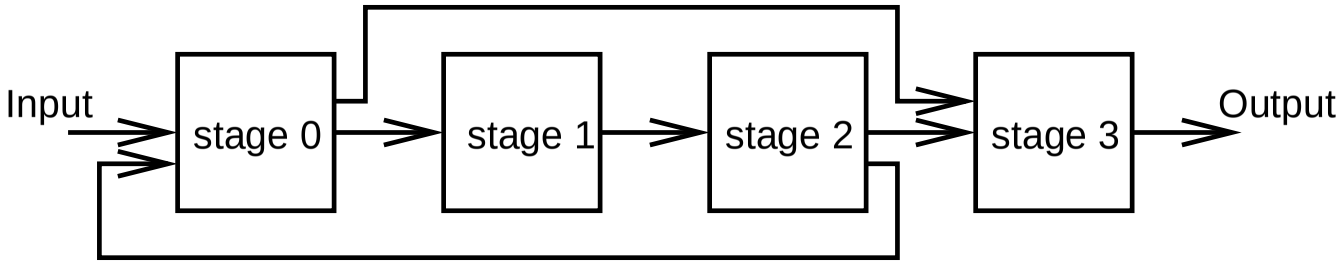
Κλιμακωτή Εκτέλεση (*Pipelining*)

- η κλιμακωτή εκτέλεση ειναι μια τεχνική υλοποίησης επεξεργαστών, όπου η εκτέλεση πολλαπλών εντολών συμπίπτει.
- η εκτέλεση των εντολών χωρίζεται σε πολλαπλά μέρη που ονομάζονται 'στάδια'.
- το κάθε στάδιο μπορεί να δουλέυει παράλληλα και ανεξάρτητα απο τα άλλα πραγματοποιώντας μέρος απο το έργο μιας εντολής.
- τα στάδια συνδέονται το ένα με το άλλο (συνήθως γραμμικά).
- έτσι, η εκτέλεση πολλαπλών εντολών συμπίπτει στο χρόνο.

Γραμμική και Μη-γραμμική κλιμάκωση



Linear Pipeline



Non-Linear Pipeline

Παράμετροι μιας κλιμάκωσης

Οι βασικοί παράμετροι μιας κλιμάκωσης ειναι οι παρακάτω:

- ροή (*throughput*): συχνότητα εισόδου/εξόδου εντολών.
- κύκλος μηχανής (*machine cycle*): χρόνος που χρειάζεται για να μετακινηθεί μια εντολή απο το ένα στάδιο στο άλλο.
- συνήθως ο κύκλος μηχανής ταυτίζεται με τον κύκλο του ρολογιού.
- η ταχύτητα μιας κλιμάκωσης εξαρτάται απο την ταχύτητα του πιό αργού σταδίου της.

Παράμετροι μιας κλιμάκωσης

- σε ένα σύστημα σύγχρονο (με ρολόι) ο βασικός στόχος του σχεδιαστή είναι να ισορροπήσει τον χρόνο του κάθε σταδίου μιας κλιμάκωσης.

- σε αυτήν την περίπτωση και σε ιδανικές συνθήκες (επιβάρυνση στην καθυστέρηση του κυκλώματος ίση με μηδέν) ο χρόνος ανα εντολή θα είναι:

$$\frac{\text{Χρόνος ανα εντολή σε μη-κλιμακωτή μηχανή}}{\text{Αριθμός στάδιων της κλιμάκωσης}}$$

- η κλιμάκωση μειώνει τους κύκλους ανά εντολή, CPI.

Απλή Υλοποίηση του DLX

1. Instruction Fetch cycle (IF):

$$\text{IR} \leftarrow \text{Mem[PC]};$$
$$\text{NPC} \leftarrow \text{PC + 4};$$

2. Instruction Decode/Register Fetch (ID):

$$\text{A} \leftarrow \text{Regs}[IR_{6...10}];$$
$$\text{B} \leftarrow \text{Regs}[IR_{11...15}];$$
$$\text{Imm} \leftarrow ((IR_{16})^{16} \# \# IR_{16...31});$$

Απλή Υλοποίηση του DLX

3. Execution/effective address cycle (EX):

- Memory Reference:

ALUOutput $\leftarrow$ A + Imm;

- Register-Register ALU Instruction:

ALUOutput $\leftarrow$ A $func$ B;

- Register-Immediate ALU Instruction:

ALUOutput $\leftarrow$ A op Imm;

- Branch:

ALUOutput $\leftarrow$ NPC + Imm;
Cond $\leftarrow$ (A op 0);

Απλή Υλοποίηση του DLX

4. Memory Access/branch completion cycle (MEM):

- Memory Reference:

LMD ← Mem[ALUOutput] or
Mem[ALUOutput] ← B;

- Branch

if (cond) PC ← ALUOutput;

Απλή Υλοποίηση του DLX

5. Write-back cycle (WB):

- Register-Register ALU Instruction:

$$\text{Regs}[IR_{16\ldots20}] \leftarrow \text{ALUOutput};$$

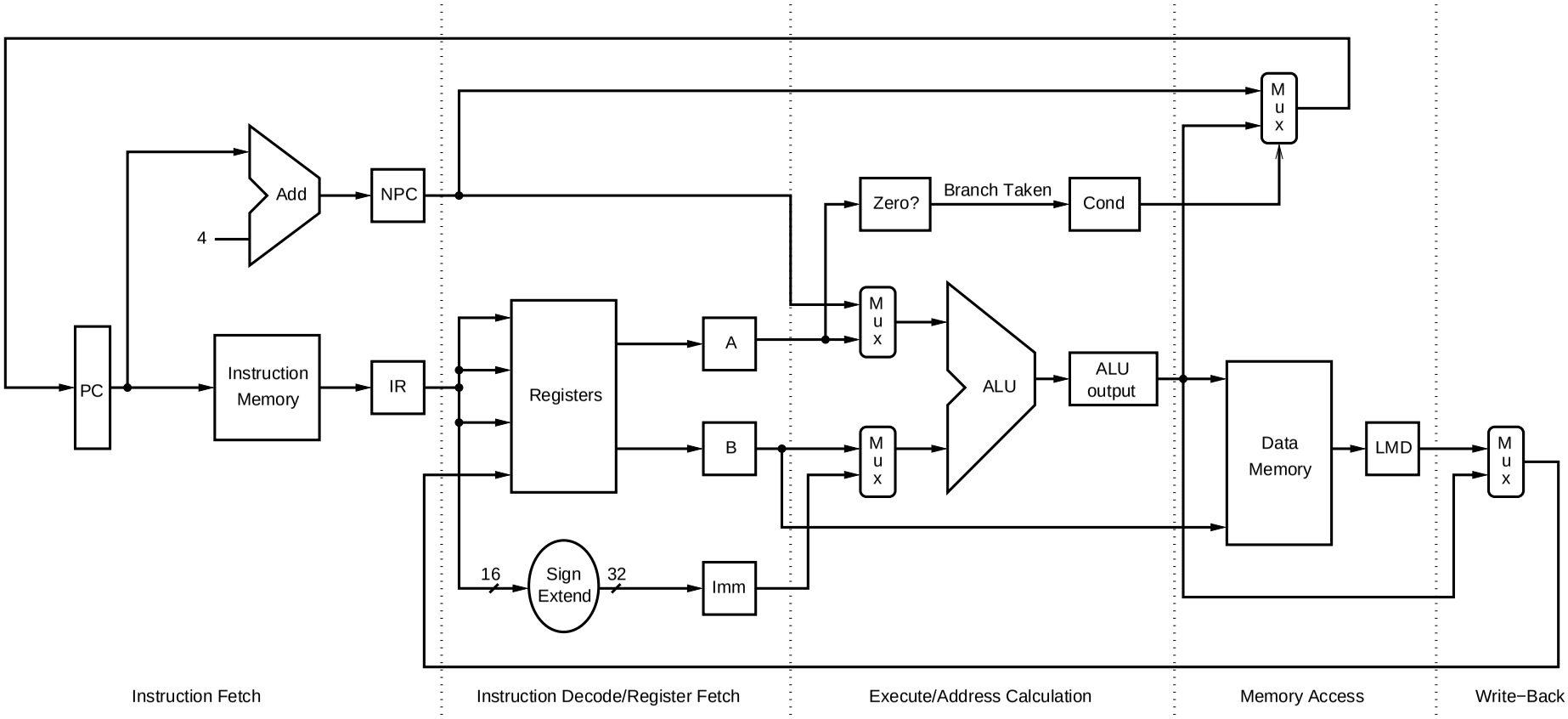
- Register-Immediate ALU Instruction:

$$\text{Regs}[IR_{11\ldots15}] \leftarrow \text{ALUOutput};$$

- Load Instruction

$$\text{Regs}[IR_{11\ldots15}] \leftarrow \text{LMD};$$

Το απλό *datapath* του DLX



Το απλό *datapath* του DLX - Παρατηρήσεις

- οι εντολές *store* και *branch* απαιτούν τέσσερις κύκλους, ενώ όλες οι άλλες απαιτούν πέντε για να ολοκληρωθούν.
- οι αριθμητικές/λογικές εντολές αδρανούν στον κύκλο MEM.
- υπάρχουν δύο αριθμητικές μονάδες ενώ θα μπορούσε να αρκέσει μία.
- υπάρχουν ξεχωριστές μνήμες για πρόγραμμα και δεδομένα.

Πόσο είναι το CPI για αυτό το μηχάνημα, αν η συχνότητα των εντολών *store* είναι 5% και των *branch* 12%;

Η κλιμάκωση του DLX

Αριθμός Κύκλου

Εντολή	1	2	3	4	5	6	7	8	9
i	IF	ID	EX	MEM	WB				
$i+1$		IF	ID	EX	MEM	WB			
$i+2$			IF	ID	EX	MEM	WB		
$i+3$				IF	ID	EX	MEM	WB	
$i+4$					IF	ID	EX	MEM	WB

- ο DLX έχει πέντε στάδια: IF, ID, EX, MEM, WB.
- αν μία εντολή αρχίζει σε κάθε κύκλο, η απόδοση της μηχανής είναι πέντε φορές της μηχανής χωρίς κλιμάκωση.

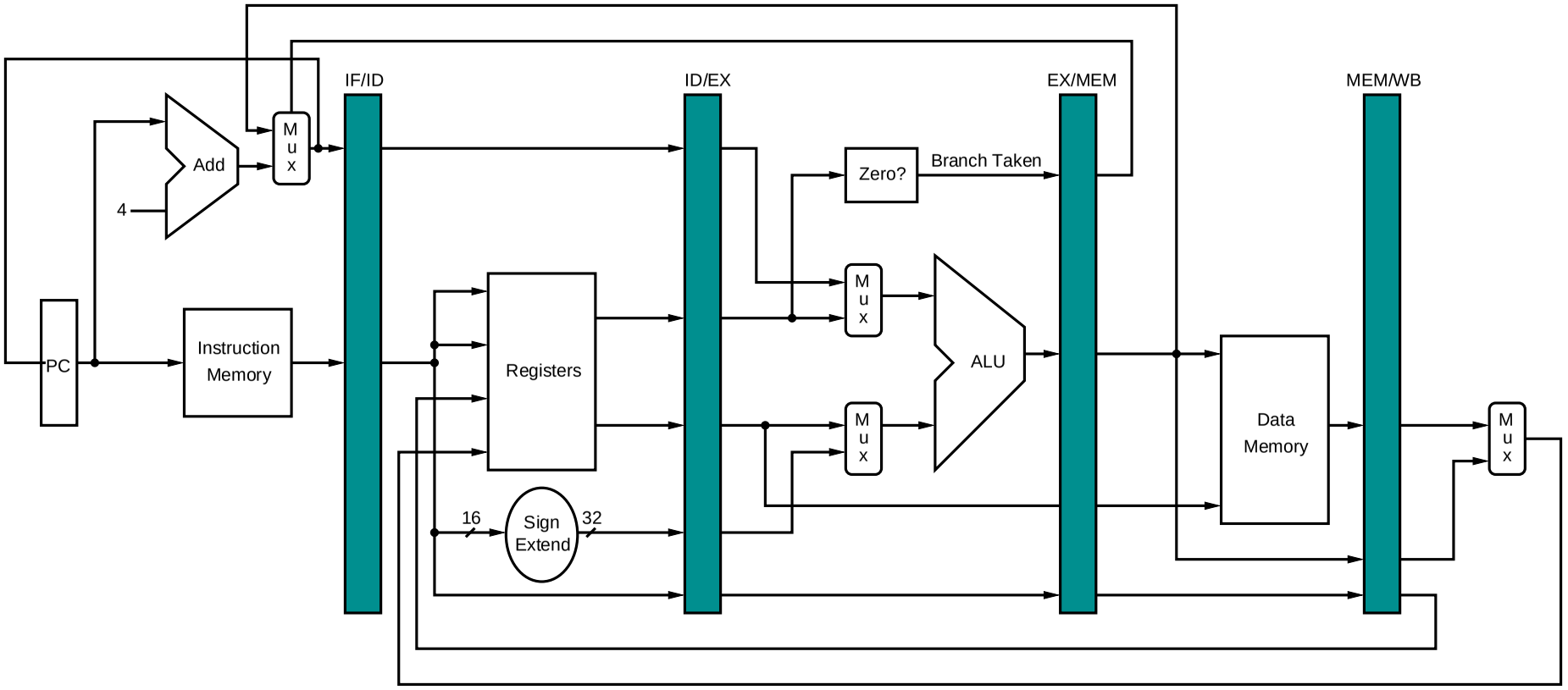
Η κλιμάκωση του DLX - Παρατηρήσεις

- για να είναι σωστή η κλιμάκωση του *datapath* πρέπει οι λειτουργίες κάθε κύκλου να είναι ανεξάρτητες μεταξύ τους και να μήν χρησιμοιποιούν κοινά στοιχεία.
- έτσι τα δεδομένα (αλλα καί ο έλεγχος) του κάθε σταδίου δέν πρέπει να συμπίπτουν.
- οι δύο μνήμες (προγράμματος και δεδομένων) επιτρέπουν ταυτόχρονη πρόσβαση στους κύκλους IF και MEM.
- και οι δύο μνήμες πρέπει να δουλεύουν πέντε φορές γρηγορότερα απο την μηχανή που δεν είχε κλιμάκωση!

Η κλιμάκωση του DLX - Παρατηρήσεις

- οι καταχωρητές ($registers$) χρησιμοποιούνται ταυτόχρονα απο δύο στάδια: ID και WB.
- τι θα συμβεί αν ο καταχωρητής που γράφεται απο το σταδίο WB ταυτόχρονα διαβάζεται απο το στάδιο ID;
- τι θα συμβεί αν υπάρξει βρόχος;

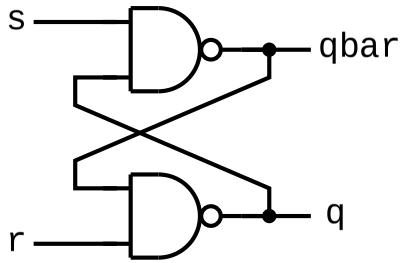
Υλοποίηση κλιμακωμένου *datapath* του DLX



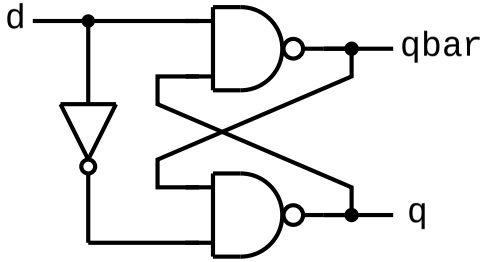
Υλοποίηση κλιμακωμένου *datapath* του DLX

- το κάθε στάδιο της κλιμάκωσης χωρίζεται απο τα άλλα με ακμοπυροδότητους καταχωρητές (*edge-triggered registers*).
- αυτοί ονομάζονται και καταχωρητές κλιμάκωσης (*pipeline registers*).
- οι καταχωρητές κλιμάκωσης μεταφέρουν σήματα ελέγχου και δεδομένα απο το ένα στάδιο στο άλλο.
- η ιδιότητα της ακμοπυροδότησης απομονώνει τα στάδια μεταξύ τους.

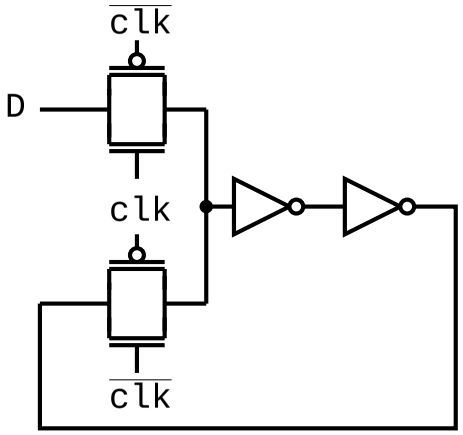
Υλοποίηση καταχωρητών



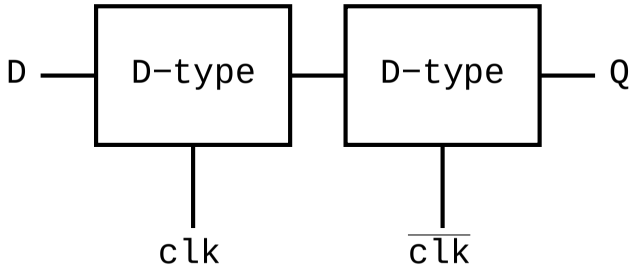
SR Latch



D Latch



CMOS +ve Latch



MS Latch – Edge–Triggered

Πρακτικά θέματα κλιμάκωσης

- η κλιμακωτή εκτέλεση αυξάνει την ροή εντολών, αλλα δεν μειώνει τον χρόνο εκτέλεσης της κάθε εντολής.
- μάλιστα αυτός αυξάνεται και θέτει όρια στο βάθος (η πλάτος) της κλιμάκωσης (λόγω εξαρτήσεων μεταξύ σταδίων).
- πρακτικά όρια επίσεις θέτονται από ανισορροπίες μεταξύ των σταδίων και το ‘πρόστιμο’ (***overhead***) της κλιμάκωσης.
- το πρόστιμο προέρχεται απο τις καθυστερήσεις των καταχωρητών και απο αποκλίσεις (***skew***) του ρολογιού.

Αλληλοεξαρτήσεις Εντολών - κίνδυνοι (*hazards*)

- υπάρχουν περιπτώσεις, που ονομάζονται κίνδυνοι (*hazards*), όπου αποτρέπουν την επόμενη εντολή να εκτελεστεί στον κύκλο/θέση που θα εκτελείτο.

- για παράδειγμα, τί θα συμβεί αν δυο εντολές εξαρτώνται, και η μία ακολουθέι την άλλη;

```
LI R2, 67
LW R1, 100(R0)
ADD R3, R1, R2
```

- η παρουσία κινδύνων/εξαρτήσεων μειώνει το κέρδος της κλιμάκωσης και εξαρτάται απο το πρόγραμμα που εκτελείται.

Αλληλοεξαρτήσεις Εντολών - κίνδυνοι (*hazards*)

Υπάρχουν τριών ειδών κίνδυνοι:

1. Δομικοί (*Structural hazards*). Λόγω σύμπτωσης δύο λειτουργιών που χρησιμοποιούν ίδιες μονάδες.

2. Δεδομένων (*Data hazards*). Όταν τα δρώμενα εντολών εξαρτώνται μεταξύ τους και η κλιμάκωση παραβιάζει την εξάρτηση.

3. Ελέγχου (*Control hazards*). Όταν αλλάζει η ροή (βρόχος).

Ο σωστός χειρισμός των κινδύνων απαιτεί την καθυστέρηση εντολών.

Δομικοί Κίνδυνοι

- συμβαίνουν όταν κάποιος συνδιασμός εντολών χρησιμοποιεί τις ίδιες μονάδες της μηχανής.
- συχνά όταν κάποιες μονάδες δεν είναι κλιμακωμένες στον ίδιο βαθμό, π.χ. μονάδες που απαιτούν πάνω απο ένα κύκλο λειτουργίας.
- επίσεις όταν μια μονάδα δεν έχει αρκετή πρόσβαση απο άλλες, π.χ. καταχωρητές με 1 θύρα εγγραφής, ενώ σε κάποιο κύκλο μπορεί να υπάρχουν δυο εγγραφές.
- σε περίπτωση δομικού κινδύνου, μία απο τις εντολές πρέπει να σταματηθεί και να περιμένει.

Παράδειγμα Δομικού Κινδύνου

Στην περίπτωση μίας μόνο θύρας μνήμης υπάρχει η πιθανότητα δομικού κινδύνου:

	1	2	3	4	5	6	7	8	9
LOAD	IF	ID	EX	**MEM**	WB				
		IF	ID	EX	MEM	WB			
			IF	ID	EX	MEM	WB		
FETCH				**IF**	ID	EX	MEM	WB	
					IF	ID	EX	MEM	WB

	1	2	3	4	5	6	7	8	9
LOAD	IF	ID	EX	**MEM**	WB				
		IF	ID	EX	MEM	WB			
			IF	ID	EX	MEM	WB		
FETCH				$stall$	**IF**	ID	EX	MEM	WB
						IF	ID	EX	MEM

Δομικοί Κίνδυνοι

Γιατι η σχεδίαση της αρχιτεκτονικής να επιτρέπει να συμβούν δομικοί κίνδυνοι;

- κόστος: είναι ακριβό να επιτρέπουμε πολλαπλές προσβάσεις σε μονάδες.
- καθυστέρηση: μονάδες που δέν υποστηρίζουν κλιμάκωση έχουν μικρότερη καθυστέρηση (*latency*), ελλείψει καταχωρητών.
- η σχεδίαση για αποφυγή δομικών κινδύνων εξαρτάται απο την συχνότητα τους (νόμος *Amdahl*).

Κίνδυνοι δεδομένων

- οι εντολές επικοινωνούν μεταξύ τους μέσω των καταχωρητών και της μνήμης.
- η κλιμάκωση αλλάζει τον χρονισμό των εντολών, εκτελώντας τις νωρίτερα.
- έτσι, αν οι εξαρτήσεις μεταξύ των εντολών δεν ικανοποιηθούν, το πρόγραμμα δεν θα εκτελεστεί σωστά.

Παράδειγμα:

```
ADD R1, R2, R3
SUB R4, R1, R5
AND R6, R1, R7
OR R8, R1, R9
XOR R10, R1, R11
```

	1	2	3	4	5	6	7	8	9
ADD **R1**, R2, R3	IF	ID	EX	MEM	**WB**				
SUB R4, **R1**, R5		IF	**ID**	EX	MEM	WB			
AND R6, **R1**, R7			IF	**ID**	EX	MEM	WB		
OR R8, **R1**, R9				IF	**ID**	EX	MEM	WB	
XOR R10, **R1**, R11					IF	**ID**	EX	MEM	WB

- το αποτέλεσμα της εντολής $ADD\ R1, R2, R3$ δεν γράφεται στον $R1$ μέχρι τον πέμπτο κύκλο.

- η επόμενη εντολή, $SUB\ R4, R1, R5$, θα προσπαθήσει να διαβάσει τον $R1$ στον τρίτο κύκλο, και οι ακόλουθες στους τρείς επόμενους.

- οι εντολές που διαβάζουν τον $R1$ πρέπει να σταματηθούν για να τρέξει το πρόγραμμα σωστά.

	1	2	3	4	5	6	7	8	9
ADD **R1**, R2, R3	IF	ID	EX	MEM	**WB**				
SUB R4, **R1**, R5		IF	$stall$	$stall$	$stall$	**ID**	EX	MEM	WB
AND R6, **R1**, R7						IF	**ID**	EX	MEM
OR R8, **R1**, R9							IF	**ID**	EX
XOR R10, **R1**, R11								IF	**ID**

- για την σωστή εκτέλεση του προγράμματος απαιτείται πρόστιμο τριών κύκλων.
- κόβοντας τους κύκλους στην μέση και γράφοντας στους καταχωρητές στον πρώτο, ενώ διαβάζοντας στο δεύτερο μισό μπορούμε να μειώσουμε το πρόστιμο σε δυο κύκλους.

	1	2	3	4	5	6	7	8	9
ADD **R1**, R2, R3	IF	ID	EX	MEM	**WB**				
SUB R4, **R1**, R5		IF	$stall$	$stall$	**ID**	EX	MEM	WB	
AND R6, **R1**, R7					IF	**ID**	EX	MEM	WB
OR R8, **R1**, R9						IF	**ID**	EX	MEM
XOR R10, **R1**, R11							IF	**ID**	EX

Προώθηση - *Forwarding*

- η τεχνική της προώθησης (ή προσπέρασης - ***bypassing***) δεδομένων λύνει το πρόβλημα των κύκλων αναμονής του αποτελέσματος.
- παρατηρώντας ότι, σε περίπτωση εξάρτησης, το αποτέλεσμα της πρώτης εντολής δεν είναι απαραίτητο μέχρι το στάδιο EX της δεύτερης μπορούμε να αποφύγουμε την αναμονή ώς εξής:

1. η έξοδος του καταχωρητή κλιμάκωσης EX/MEM (μετά το στάδιο EX) επιστρέφει (προωθούμενο αποτέλεσμα) και στη είσοδο του EX ($EX/MEM \rightarrow EX$).
2. το ίδιο κάνουμε και για της έξοδο του MEM/WB ($MEM/WB \rightarrow EX$).
3. προσθέτουμε λογική που θα διαλέγει το προωθούμενο αποτέλεσμα όταν ο καταχωρητής εγγραφής της παρούσας εντολής συμπίπτει με έναν απο τους πηγαίους των επομένων.

Προώθηση - *Forwarding*

Υλοποιώντας προώθηση δεδομένων και γράφοντας/διαβάζοντας στα μισά του κύκλου αποφεύγουμε οποιαδήποτε πρόστιμο:

	1	2	3	4	5	6	7	8	9
ADD **R1**, R2, R3	*IF*	*ID*	*EX*	*MEM*	**WB**				
SUB R4, **R1**, R5		*IF*	**ID**	*EX*	*MEM*	*WB*			
AND R6, **R1**, R7			*IF*	**ID**	*EX*	*MEM*	*WB*		
OR R8, **R1**, R9				*IF*	**ID**	*EX*	*MEM*	*WB*	
XOR R10, **R1**, R11					*IF*	**ID**	*EX*	*MEM*	*WB*

Τύποι Κινδύνων Δεδομένων

Οι κίνδυνοι δεδομένων μπορούν να χωριστούν σε τρείς περιπτώσεις και ονομάζονται με την σειρά που πρέπει να ακολουθηθεί απο την κλιμάκωση.

Για δύο εντολές i και j, όπου η i είναι πριν την j οι ακόλουθοι κίνδυνοι είναι πιθανοί:

- RAW $(read\ after\ write)$ – η j προσπαθεί να διαβάσει τα πηγαία πρίν τα γράψει η j, έτσι διαβάζει τα παλιά δεδομένα.

- WAW $(write\ after\ write)$ – η j γράφει στο δρώμενο της πρίν απο την i, οι εγγραφές γίνονται με λάθος σειρά.

- WAR $(write\ after\ read)$ – η j γράφει στο δρώμενο της πρίν αυτό διαβαστεί απο την i, έτσι η τελευταία διαβάζει τη νέα τιμή.

Κίνδυνοι Δεδομένων στον DLX

οι κίνδυνοι WAW και WAR δεν υφίστανται στον DLX.

- κίνδυνος WAW υπάρχει σε κλιμακώσεις όπου υπάρχουν πολλαπλά στάδια εγγραφής (WB), ή αφήνουν εντολές να προσπεράσουν στάδια:

LW **R1**, 0(R2)	IF	ID	EX	$MEM1$	$MEM2$	**WB**
ADD **R1**, R2, R3		IF	ID	EX	**WB**	

- κίνδυνος WAR υπάρχει σε κλιμακώσεις όπου η εγγραφή των δεδομένων μπορεί να προσπεράσει την ανάγνωση:

SW 0(R1), **R2**	IF	ID	EX	$MEM1$	**MEM2**	WB
ADD **R2**, R3, R4		IF	ID	EX	**WB**	

Απαιτούμενη Αναμονή

- όλες οι εξαρτήσεις δεν μπορούν να χειριστούν με προώθηση δεδομένων:

```
LW R1, 0(R2)
SUB R4, R1, R5
AND R6, R1, R7
OR R8, R1, R9
```

	1	2	3	4	5	6	7	8
LW **R1**, 0(R2)	IF	ID	EX	**MEM**	WB			
SUB R4, **R1**, R5		IF	**ID**	EX	MEM	WB		
AND R6, **R1**, R6			IF	**ID**	EX	MEM	WB	
OR R8, **R1**, R9				IF	**ID**	EX	MEM	WB

Εξάρτηση Μνήμης

- τα δεδομένα δεν είναι απαραίτητα πρίν το στάδιο EX.
- όμως ακόμα και αν τα προωθήσουμε απο το MEM/WB στο EX απαιτείται αναμονή ενός κύκλου:

	1	2	3	4	5	6	7	8
LW **R1**, 0(R2)	IF	ID	EX	**MEM**	WB			
SUB R4, **R1**, R5		IF	ID	$stall$	**EX**	MEM	WB	
AND R6, **R1**, R6			IF	$stall$	ID	**EX**	MEM	WB
OR R8, **R1**, R9				$stall$	IF	**ID**	EX	MEM

- για να υλοποιηθεί ο κύκλος αναμονής πρέπει να προσθέσουμε λογική στην αρχιτεκτονική που να αντιλαμβάνεται την εξάρτηση.
- η λογική αυτή ονομάζεται κλείδωμα κλιμάκωσης (***pipeline interlock***) και σταματάει την ροή στην κλιμάκωση, απο την εντολή που χρειάζεται δεδομένα, μέχρι αυτά να είναι διαθέσιμα.

Δρομολόγηση Εντολών (*scheduling*)

- μερικοί συνδιασμοί εντολών που προκαλούν αναμονή είναι συχνοί.
- π.χ. ο κώδικας που παράγεται για προτάσεις του τύπου $A = B + C$.

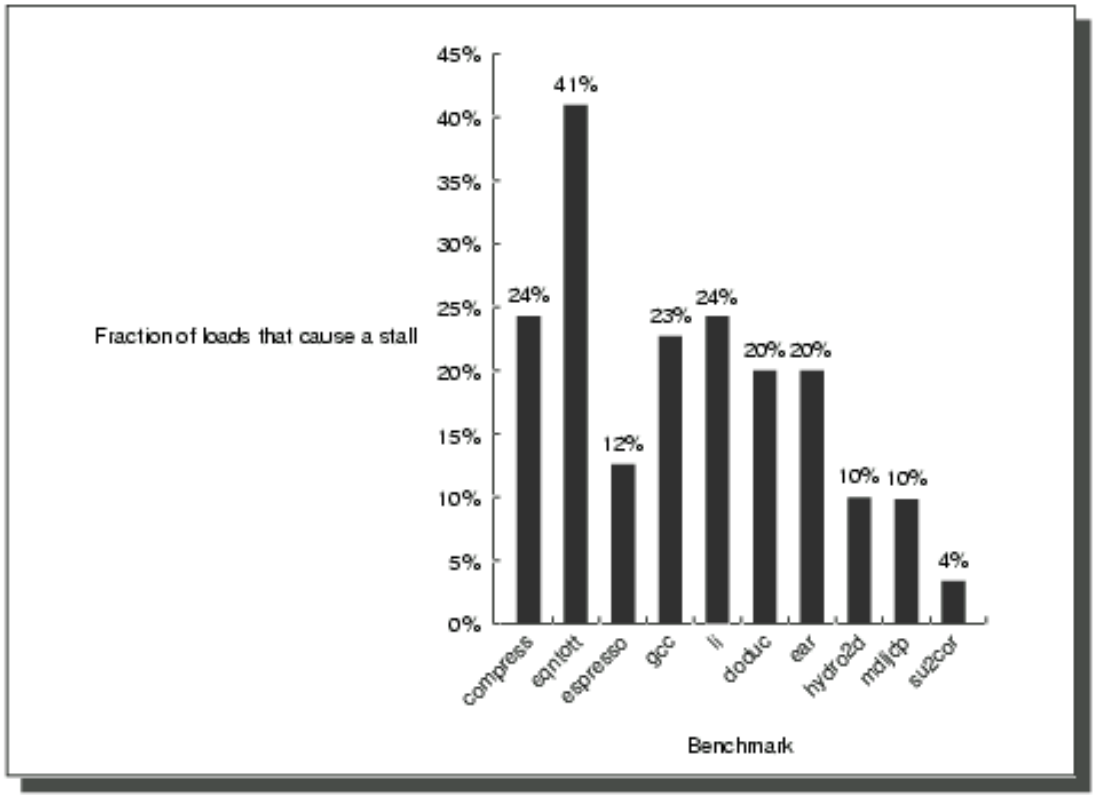
	1	2	3	4	5	6	7	8	9
LW R1, B	IF	ID	EX	MEM	WB				
LW R2, C		IF	ID	EX	MEM	WB			
ADD R3, R1, R2			IF	ID	*stall*	EX	MEM	WB	
SW A, R3				IF	*stall*	ID	EX	MEM	WB

- μπορούμε να παρεμβάλουμε εντολές με σκοπό να απαληφεί το σταμάτημα.
- στο παράδειγμα μπορούμε να παρεμβάλουμε μια εντολή ανάμεσα στις: $LW\ R2, C$ και $ADD\ R3, R1, R2$.

Υλοποίηση Σταματήματος/Προώθησης

- για να σταματήσουμε την κλιμάκωση πρέπει να υλοποιήσουμε λογική που αναλύει τις εξαρτήσεις μεταξύ των εντολών.
- πρέπει να συγκρίνουμε τον καταχωρητή εγγραφής μιας εντολής με τους πηγαίους των εντολών που την ακολουθούν.
- ανάλογα με την περίσταση, πρέπει να σταματήσουμε κάποιο στάδιο ή να προωθήσουμε δεδομένα.

Καθυστερήσεις λόγω εντολών Μνήμης στον DLX



Πιθανές Εξαρτήσεις

Περίσταση	Παράδειγμα κώδικα	Δράση
καμμία εξάρτηση	LW **R1**, 45(R2) ADD R5, R6, R7 SUB R8, R6, R7 OR R9, R6, R7	Κανένας κίνδυνος μια και δεν υπάρχει εξάρτηση στον $R1$ στις επόμενες τρίς εντολες.
εξάρτηση που απαιτεί σταμάτημα	LW **R1**, 45(R2) ADD R5, **R1** , R7 SUB R8, R6, R7 OR R9, R6, R7	Συγκριτές αναγνωρίζουν την χρήση του $R1$ στο ADD και σταματάνε το ADD πριν το στάδιο EX.
εξάρτηση που απαιτεί προώθηση	LW **R1**, 45(R2) ADD R5, R6, R7 SUB R8, **R1**, R7 OR R9, R6, R7	Συγκριτές αναγνωρίζουν την χρήση του $R1$ στο SUB και προωθούν το αποτέλεσμα του MEM στο EX του SUB.
εξάρτηση με προσβάσεις σε σειρά	LW **R1**, 45(R2) ADD R5, R6, R7 SUB R8, R6, R7 OR R9, **R1**, R7	Καμία πράξη δεν χρειάζεται λογω της σειράς ανάγνωσης/ εγγραφής δεδομένων στα στάδια ID καί WB.

Υλοποίηση Προώθησης - 1

Καταχωρητής που περιέχει το πηγαίο	Τύπος πηγαίας εντολής	Τύπος λαμβάνουσας εντολής	Προορισμός προωθούμενου αποτελεσματος	Συνθήκη
EX/MEM	Κατ.-Κατ.-ALU	Κατ.-Κατ.-ALU, Κατ.-άμεσο-ALU μνήμης, βρόχος.	Πάνω είσοδος ALU	$EX/MEM.IR_{16..20} == ID/EX.IR_{6..10}$
EX/MEM	Κατ.-Κατ.-ALU	Κατ.-Κατ.-ALU,	Κάτω είσοδος ALU	$EX/MEM.IR_{16..20} == ID/EX.IR_{11..15}$
MEM/WB	Κατ.-Κατ.-ALU	Κατ.-Κατ.-ALU, Κατ.-άμεσο-ALU μνήμης, βρόχος.	Πάνω είσοδος ALU	$MEM/WB.IR_{16..20} == ID/EX.IR_{6..10}$
MEM/WB	Κατ.-Κατ.-ALU	Κατ.-Κατ.-ALU,	Κάτω είσοδος ALU	$MEM/WB.IR_{16..20} == ID/EX.IR_{11..15}$

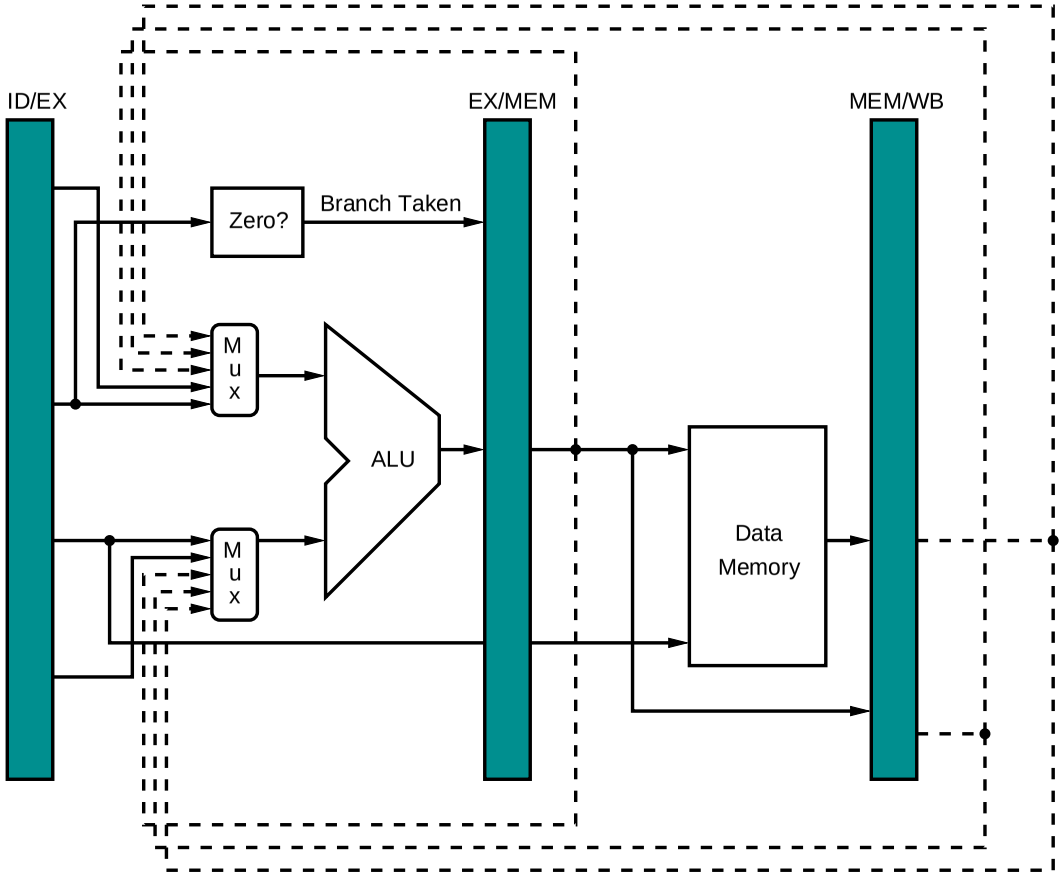
Υλοποίηση Προώθησης - 2

Καταχωρητής που περιέχει το πηγαίο	Τύπος πηγαίας εντολής	Τύπος λαμβάνουσας εντολής	Προορισμός προωθούμενου αποτελεσματος	Συνθήκη
EX/MEM	Κατ.-άμεσο-ALU	Κατ.-Κατ.-ALU, Κατ.-άμεσο-ALU μνήμης, βρόχος.	Πάνω είσοδος ALU	$EX/MEM.IR_{11..15} == ID/EX.IR_{6..10}$
EX/MEM	Κατ.-άμεσο-ALU	Κατ.-Κατ.-ALU,	Κάτω είσοδος ALU	$EX/MEM.IR_{11..15} == ID/EX.IR_{11..15}$
MEM/WB	Κατ.-άμεσο-ALU	Κατ.-Κατ.-ALU, Κατ.-άμεσο-ALU μνήμης, βρόχος.	Πάνω είσοδος ALU	$MEM/WB.IR_{11..15} == ID/EX.IR_{6..10}$
MEM/WB	Κατ.-άμεσο-ALU	Κατ.-Κατ.-ALU,	Κάτω είσοδος ALU	$MEM/WB.IR_{11..15} == ID/EX.IR_{11..15}$

Υλοποίηση Προώθησης - 3

Καταχωρητής που περιέχει το πηγαίο	Τύπος πηγαίας εντολής	Τύπος λαμβάνουσας εντολής	Προορισμός προωθούμενου αποτελεσματος	Συνθήκη
MEM/WB	*Load*	Κατ.-Κατ.-ALU, Κατ.-άμεσο-ALU μνήμης, βρόχος.	Πάνω είσοδος ALU	$MEM/WB.IR_{11..15} == ID/EX.IR_{6..10}$
MEM/WB	*Load*	Κατ.-Κατ.-ALU,	Κάτω είσοδος ALU	$MEM/WB.IR_{11..15} == ID/EX.IR_{11..15}$

Υλοποίηση Προώθησης - *Datapath*



Κίνδυνοι Ελέγχου

- στην περίπτωση εντολής βρόχου ο PC θα αλλάξει η όχι.
- αυτό εξαρτάται απο την προυπόθεση του βρόχου.
- η τιμή του PC δεν μπορεί να αλλαγεί πριν το τέλος του σταδίου MEM, μια και τότε ειναι διαθέσιμη η καινούργια τιμή του.
- η πιό απλή εκτέλεση βρόχων σταματάει την κλιμάκωση μέχρι η καινούργια τιμή του PC να είναι γνωστή.

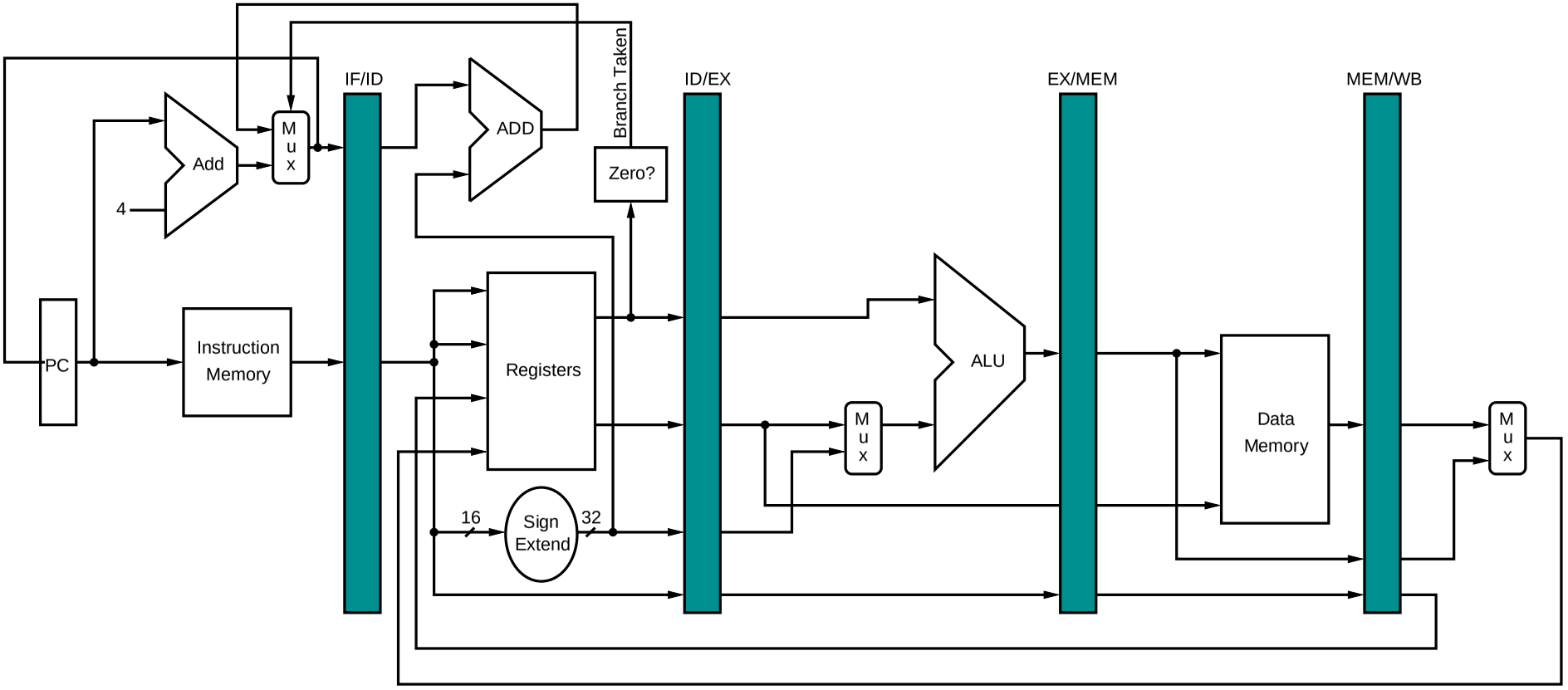
Σταμάτημα λόγω βρόχου

	1	2	3	4	5	6	7	8	9
Βρόχος	IF	ID	EX	MEM	WB				
Ακόλουθη βρόχου		IF	$stall$	$stall$	IF	ID	EX	MEM	WB
Ακόλουθη + 1						IF	ID	EX	MEM
Ακόλουθη + 2							IF	ID	EX
Ακόλουθη + 3								IF	ID

- το κόστος στην απόδοση είναι τρείς εντολές: δύο κύκλοι αναμονής και επανάληψη του στάδιου IF.
- το υψηλό κόστος βρόχων πρέπει να μειωθεί μια και είναι πολύ συνηθισμένοι.
- για να μειωθεί πρέπει:

1. να γνωρίζουμε νωρίτερα το αποτέλεσμα του βρόχου (αν ακολουθείται η όχι).
2. να υπολογίσουμε νωρίτερα την τιμή του PC, για το αν ακολουθηθεί.

Βελτίωση Βρόχων



Βελτίωση Βρόχων

- δύο αλλαγές στην οργάνωση της κλιμάκωσης.
- ο υπολογισμός της διεύθυνσης του βρόχου και της προυπόθεσης μεταβιβάστικαν στο στάδιο ID.
- στο αρχικό σχέδιο αυτό γινόταν στο στάδιο EX.
- ο PC γράφεται στο στάδιο IF με την νέα τιμή που υπολογίστηκε στο ID, ή με $PC + 4$.
- στο αρχικό σχέδιο ο PC γραφόταν απο τον καταχωρητή EX/MEM στο στάδιο MEM.

Βελτίωση Βρόχων

	1	2	3	4	5	6	7	8	9
Βρόχος	IF	ID	EX	MEM	WB				
Ακόλουθη βρόχου		IF	IF	ID	EX	MEM	WB		
Ακόλουθη + 1				IF	ID	EX	MEM	WB	
Ακόλουθη + 2					IF	ID	EX	MEM	WB
Ακόλουθη + 3						IF	ID	EX	MEM

- κάνοντας αυτές τις αλλαγές η αναμονή λόγω βρόχου πέφτει στον έναν κύκλο.
- το πρόστιμο των βρόχων σχετίζεται άμεσα με την σχεδίασης της κλιμάκωσης.
- π.χ. άν η κλιμάκωση είχε ξεχωριστά στάδια ID και RF ($Register\ Fetch$ - ανάγνωσης καταχωρητών), τότε το πρόστιμο θα ήταν ένας κύκλος ακόμα.

Τρόποι Μείωσης Κόστους Βρόχων

- ο πιό απλός τρόπος χειρισμού βρόχων είναι το σταμάτημα/πάγωμα της κλιμάκωσης.
- έτσι, οι ακόλουθες εντολές κρατώνται ή σβήνωνται όταν το αποτέλεσμα του βρόχου γίνει γνωστό.
- αυτός ο τρόπος χειρισμού παράγει μια σταθερή καθυστέρηση για κάθε βρόχο.
- ένας πιό πολύπλοκος τρόπος χειρισμού βρόχων ειναι η πρόγνωση του αποτελέσματος.
- με αυτήν την μέθοδο συνεχίζουμε την λειτουργία της κλιμάκωσης βασισμένοι στην πρόγνωση του βρόχου.

Πρόγνωση-μή-ακολούθησης

- η πρόγνωση-μή-ακολούθησης υποθέτει οτι ο βρόχος δέν θα ακολουθηθεί.
- έτσι η κλιμάκωση συνεχίζει να διοχετεύεται με εντολές.
- άν ο βρόχος τελικά ακολουθηθεί, τότε πρέπει να ακυρώσουμε την επόμενη εντολή καθαρίζοντας τους καταχωρητές κλιμάκωσης.
- χρειάζεται προσοχή έτσι ώστε η κατάσταση της μηχανής να μήν αλλάξει μέχρι το αποτέλεσμα του βρόχου να είναι γνωστό (π.χ. τιμές καταχωρητών).

Πρόγνωση-μή-ακολούθησης

Βρόχος που δέν ακολουθήται:

	1	2	3	4	5	6	7	8	9
Βρόχος	IF	ID	EX	MEM	WB				
Ακόλουθη βρόχου		IF	ID	EX	MEM	WB			
Ακόλουθη + 1			IF	ID	EX	MEM	WB		
Ακόλουθη + 2				IF	ID	EX	MEM	WB	
Ακόλουθη + 3					IF	ID	EX	MEM	WB

Βρόχος που ακολουθήται:

	1	2	3	4	5	6	7	8	9
Βρόχος	IF	ID	EX	MEM	WB				
Ακόλουθη βρόχου		IF	κενό	κενό	κενό	κενό			
Προορισμός Βρόχου			IF	ID	EX	MEM	WB		
Προορισμός Βρόχου + 1				IF	ID	EX	MEM	WB	
Προορισμός Βρόχου + 2					IF	ID	EX	MEM	WB

Πρόγνωση-ακολούθησης

- η αντίστροφη μέθοδος είναι η πρόγνωση-ακολούθησης.
- υποθέτει οτι ο βρόχος θα ακολουθηθεί.
- στον DLX αυτή η μέθοδος είναι ανούσια μιά και η δίευθυνση προορισμού του βρόχου δεν είναι γνωστή πρίν και το αποτέλεσμα του.

Καθυστέρημένος Βρόχος

- η μέθοδος καθυστέρησης βρόχου καθυστερεί την εκτέλεση του βρόχου εκτελώντας εντολές μετα απο αυτόν.

π.χ. για καθυστέρηση ν εντολών:

βρόχος
ακόλουθη εντολή
ακόλουθη εντολή + 1
.
.
.
ακόλουθη εντολή + ν - 1
προορισμός βρόχου

Καθυστερημένος Βρόχος

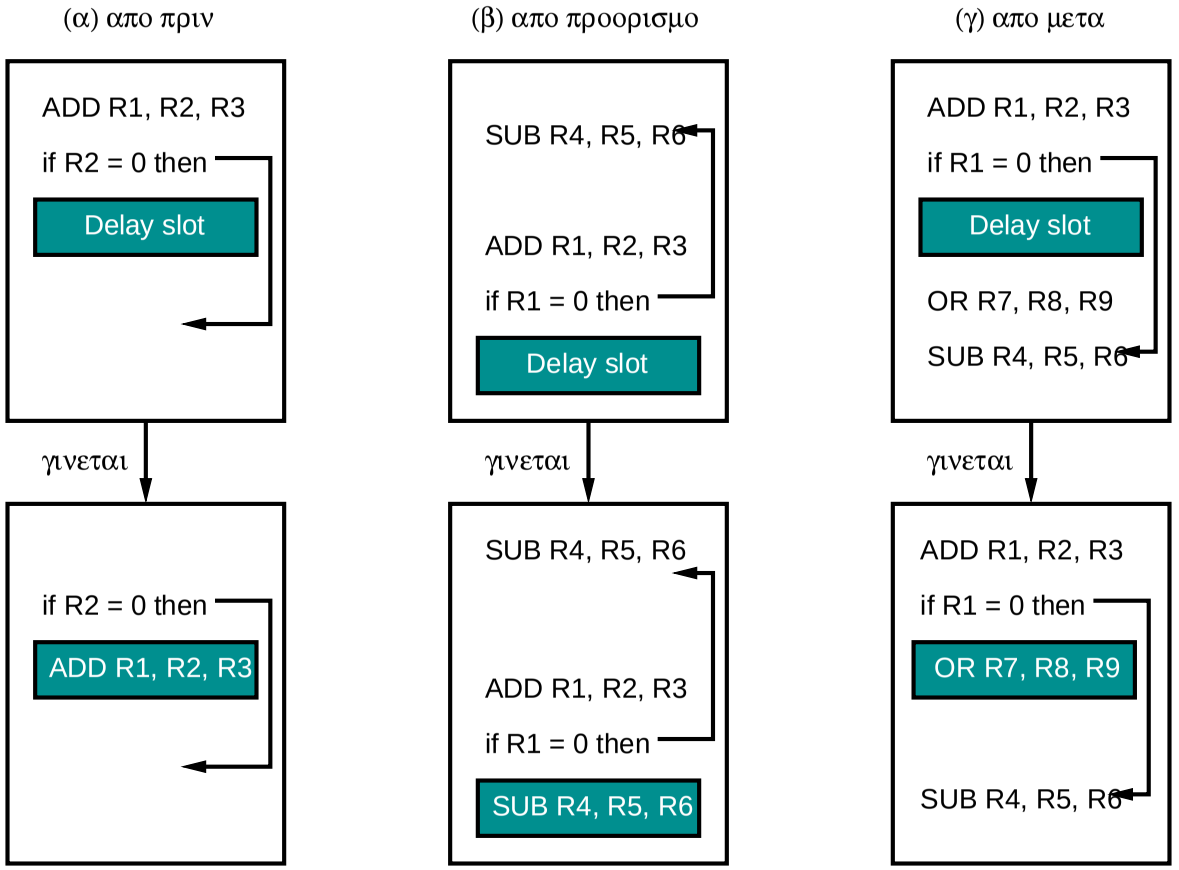
Βρόχος που δέν ακολουθήται:

	1	2	3	4	5	6	7	8	9
Βρόχος	IF	ID	EX	MEM	WB				
Εντολή καθυστέρησης βρόχου		IF	ID	EX	MEM	WB			
Ακόλουθη			IF	ID	EX	MEM	WB		
Ακόλουθη + 1				IF	ID	EX	MEM	WB	
Ακόλουθη + 2					IF	ID	EX	MEM	WB

Βρόχος που ακολουθήται:

	1	2	3	4	5	6	7	8	9
Βρόχος	IF	ID	EX	MEM	WB				
Εντολή καθυστέρησης βρόχου		IF	ID	EX	MEM	WB			
Προορισμός Βρόχου			IF	ID	EX	MEM	WB		
Προορισμός Βρόχου + 1				IF	ID	EX	MEM	WB	
Προορισμός Βρόχου + 2					IF	ID	EX	MEM	WB

Καθυστερημένος Βρόχος - Δρομολόγηση



Καθυστερημένος Βρόχος - Δρομολόγηση

- η καλύτερη περίπτωση για δρομολόγηση είναι η (α), όπου η εντολή ειναι ανεξάρτητη.
- οι (β) και (γ) χρησιμοποιούνται όταν η (α) είναι αδύνατη.
- η χρήση του $R1$ στις (β) και (γ) απαγορεύει την μετακίνηση της ADD μετά τον βρόχο.
- στην (β) η εντολή απο τον προορισμό αντιγράφεται.
- στις περιπτώσεις (β) και (γ) πρέπει το πρόγραμμα να εκτελείται σωστά όποια η κατάληξη των βρόχων.

Καθυστερημένος Βρόχος - Δρομολόγηση

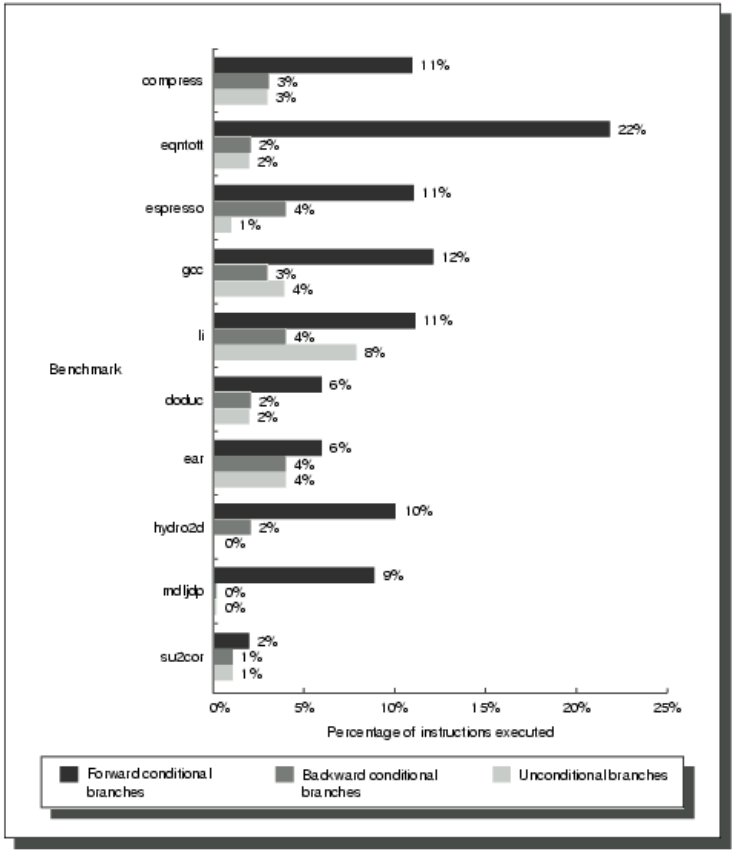
Στρατηγική Δρομολόγησης	Απαίτηση	Αυξάνει Απόδοση όταν...
(α) Απο πρίν	Ο βρόχος δεν εξαρτάται απο την μετακινούμενη εντολή.	Πάντα.
(β) Απο προορισμό	Πρέπει η εντολή που θα μετακινηθεί να μήν επηρεάζει το πρόγραμμα αν ο βρόχος δεν ακολουθηθεί.	Όταν ο βρόχος ακολουθηθεί. Μεγαλώνει πιθανώς το πρόγραμμα.
(γ) Απο μετά	Πρέπει η εντολή που θα μετακινηθεί να μήν επηρεάζει το πρόγραμμα αν ο βρόχος δεν ακολουθηθεί.	Όταν ο βρόχος δεν ακολουθηθεί.

- Η δρομολόγηση καθυστερήσεων βρόχων εξαρτάται απο: (1) τους περιορισμούς στις εντολές που μετακινούνται και...
- (2) την ικανότητα πρόγνωσης της κατεύθυνσης βρόχων κατά την κατασκευή (*compilation*) του προγράμματος.

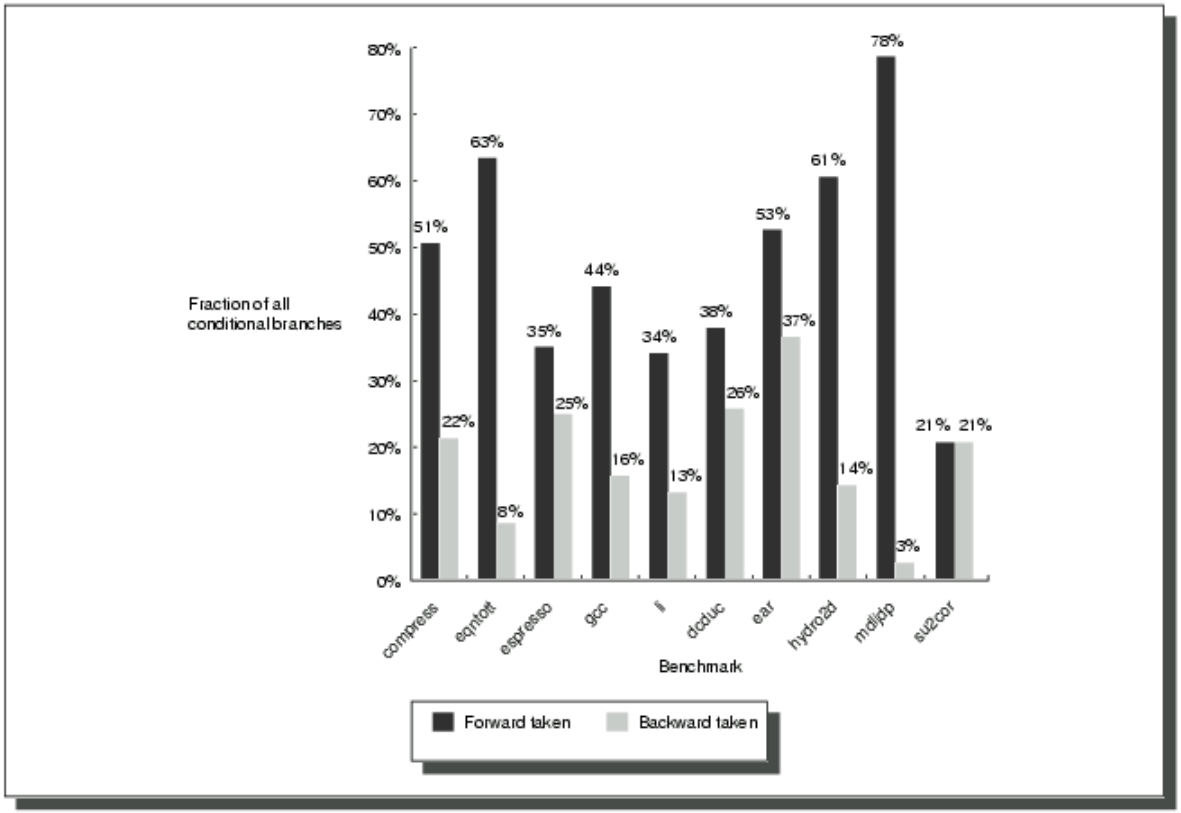
Ακυρωτικοί Βρόχοι

- για να βελτιωθεί η ικανότητα να γεμίσουν οι καθυστερήσεις βρόχων αρχιτεκτονικές προσφέρουν ενα ακυρωτικό βρόχο.
- ο ακυρωτικός βρόχος περιλαμβάνει την κατεύθυνση που προγνωείται.
- η ροή εντολών στην αρχιτεκτονική ακολουθεί την κατεύθυνση πρόγνωσης.
- αν αυτή είναι λάθος, τότε η εντολή καθυστέρησης μετατρέπεται σε NOP.
- επιτρέπουν τον μεταγλωτιστή (*compiler*) να χρησιμοποιήσει τις δρομολογήσεις (β) και (γ) χωρίς να ισχύουν οι απαιτήσεις τους.

Συμπεριφορά Βρόχων σε προγράμματα



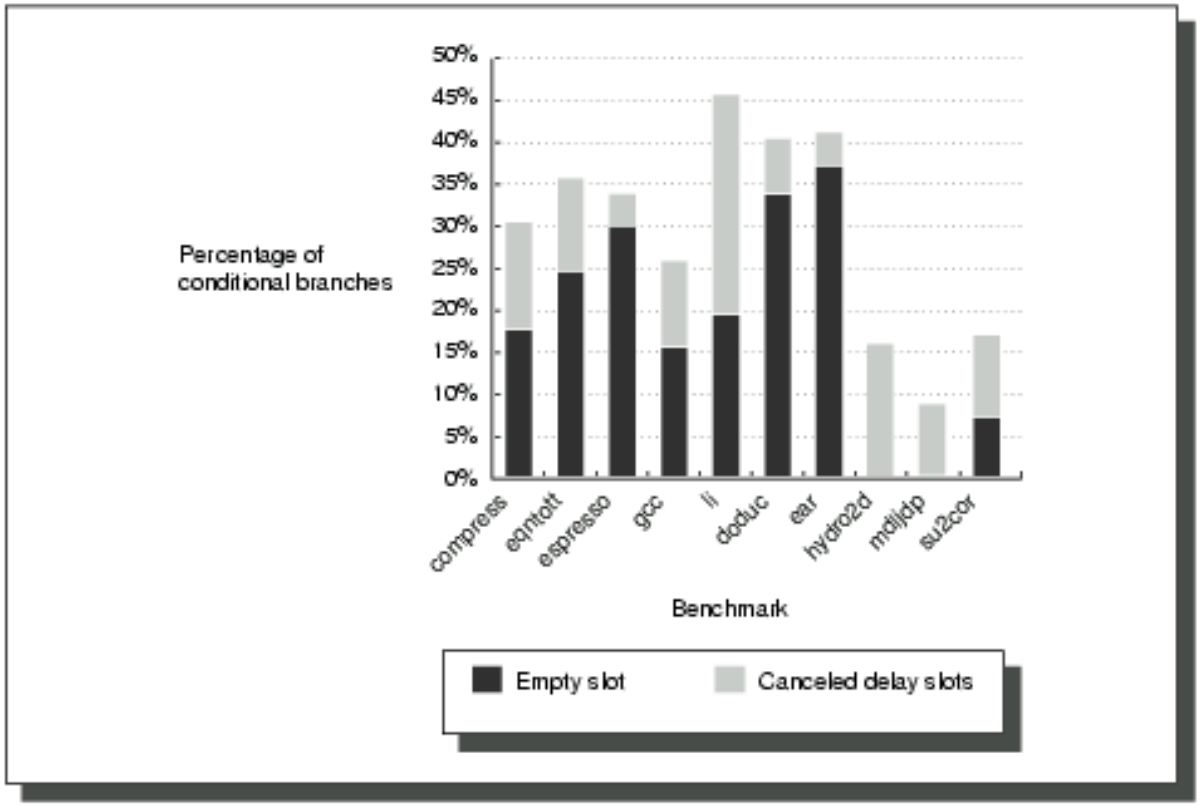
Συμπεριφορά Βρόχων σε προγράμματα



Καθυστερημένοι και Ακυρωτικοί Βρόχοι στον DLX

Benchmark	% conditional branches	% conditional branches with empty slots	% conditional branches that are cancelling	% cancelling branches that are cancelled	% branches with cancelled delay slots	Total % branches with empty or cancelled delay slot
compress	14%	18%	31%	43%	13%	31%
eqntott	24%	24%	50%	24%	12%	36%
espresso	15%	29%	19%	21%	4%	33%
gcc	15%	16%	33%	34%	11%	27%
li	15%	20%	55%	48%	26%	46%
Integer average	17%	21%	38%	34%	13%	35%
doduc	8%	33%	12%	62%	8%	41%
ear	10%	37%	36%	14%	5%	42%
hydro2d	12%	0%	69%	24%	16%	17%
mdljdp2	9%	0%	86%	10%	8%	8%
su2cor	3%	7%	17%	57%	10%	17%
FP average	8%	16%	44%	34%	9%	25%
Overall average	12%	18%	41%	34%	11%	30%

Καθυστερημένοι και Ακυρωτικοί Βρόχοι στον DLX



Μείωσης Απόδοσης λόγω βρόχων

- η απόδοση της κλιμάκωσης πέφτει λόγω της καθυστέρησης των βρόχων.
- έτσι στην ιδανική περίπτωση όπου το οι κύκλοι ανα εντολή είναι 1 (CPI=1), έχουμε:

$$\text{επιτάχυνση κλιμάκωσης} = \frac{\text{μήκος κλιμάκωσης}}{1 + \text{κύκλοι καθυστέρησης λόγω βρόχων}}$$

όπου: κύκλοι καθυστέρησης λόγω βρόχων = συχνότητα βρόχων . καθυστέρηση βρόχου
άρα:

$$\text{επιτάχυνση κλιμάκωσης} = \frac{\text{μήκος κλιμάκωσης}}{1 + \text{συχνότητα βρόχων . καθυστέρηση βρόχου}}$$

Μείωσης Απόδοσης λόγω βρόχων

- χρησιμοποιώντας τις μετρήσεις για τον DLX μπορούμε να συμπεράνουμε την επίδραση των βρόχων στην απόδοση.

- οι μετρήσεις που απαιτούνται ειναι: συχνότητες βρόχων στο πρόγραμμα, συχνότητα βρόχων που ακολουθούνται και την ικανότητα χρήσης της εντολής καθυστέρησης.

Δρομολόγηση Βρόχων	Ποινή - εξ-βρόχου		Ποινή ανεξ-βρόχου	Μέση ποινή - ανά βρόχο		CPI - με βρόχους	
	Ακ.	Ρητ.		Ακ.	Ρητ.	Ακ.	Ρητ.
Σταμάτημα	1.00	1.00	1.00	1.00	1.00	1.17	1.15
Πρόγν. ακ.	1.00	1.00	1.00	1.00	1.00	1.17	1.15
Πρόγν. μή-ακ.	0.62	0.70	1.00	0.69	0.74	1.12	1.11
Καθ. βρ.	0.35	0.25	0.00	0.30	0.21	1.06	1.03

Στατική Πρόγνωση Βρόχων

- η δρομολόγιση των εντολών για αποδοτική χρήση καθυστερημένων βρόχων εξαρτάται απο το αποτέλεσμα των βρόχων.

π.χ.:

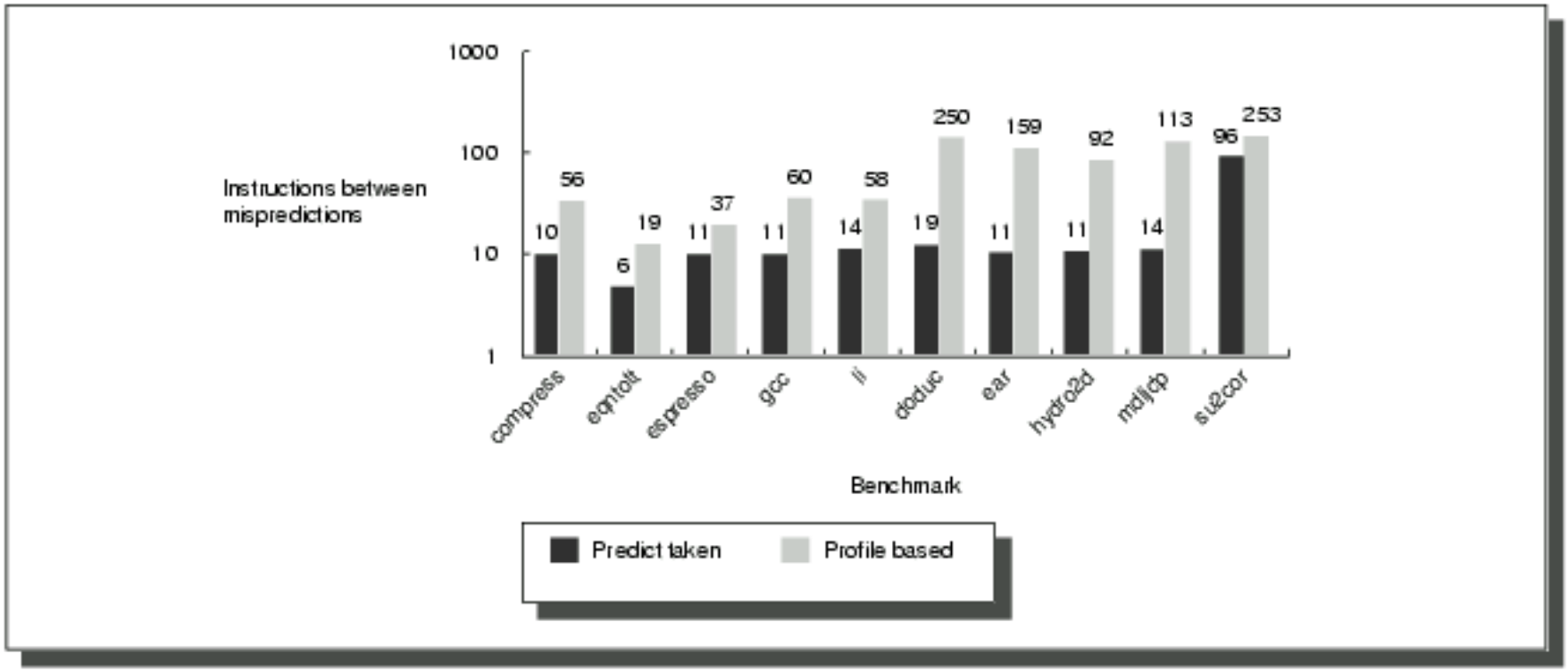
```
   LW R1, 0(R2)
   SUB R1, R1, R3 <-
   BEQZ R1, L
   OR R4, R5, R6  <-
   ADD R10, R4, R5
L: ADD R7, R8, R9 <-
```

- ανάλογα με την πιθανότητα κατάληξης του βρόχου πρέπει να γίνει η δρομολόγηση εντολών (στην μεταγλώτιση) για να αποφευχθεί αναμονή λόγω του LW.

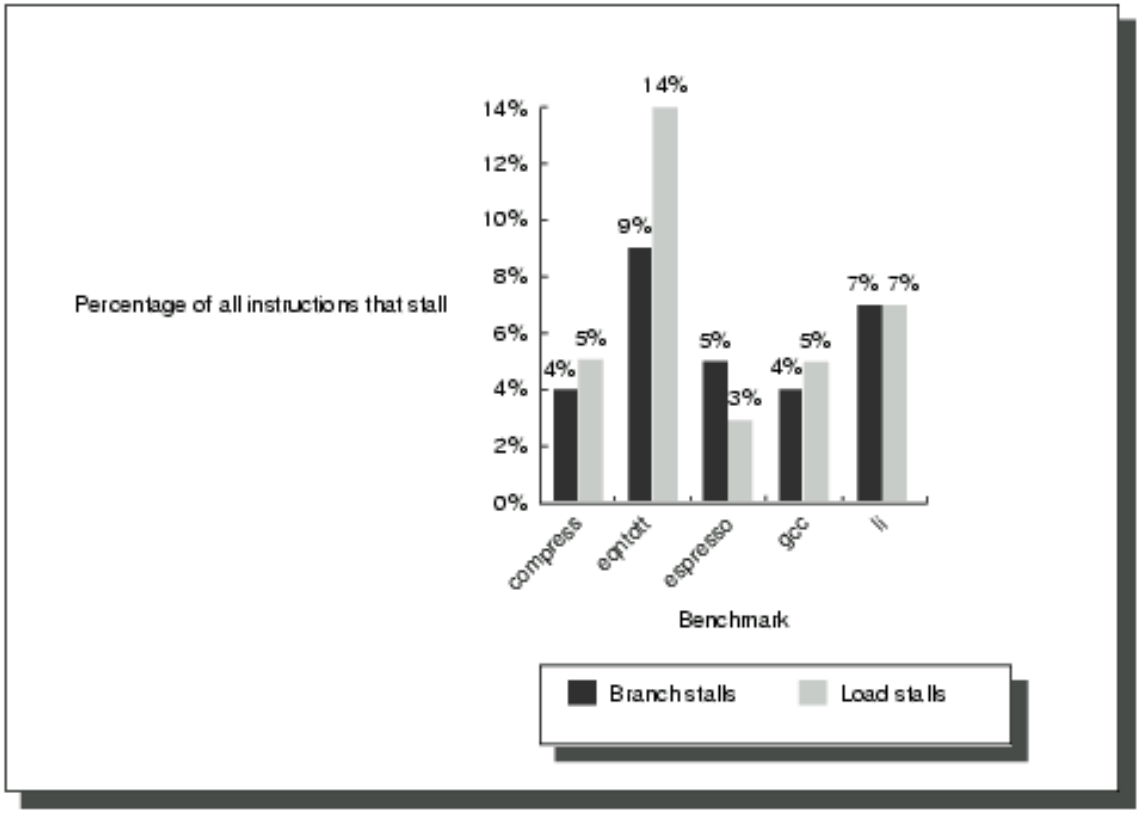
Στατική Πρόγνωση Βρόχων

- δύο ειδών μέθοδοι για στατική (απο τον μεταγλωτιστή) πρόγνωση βρόχων.
- ευριστικές μεθόδοι· σύμφωνα με την συμπεριφορά γενικών προγραμμάτων.
- π.χ. όλοι οι βρόχοι προγνωούνται οτι θα ακολουθηθούν.
- ή όλοι οι βρόχοι προς τα πίσω προγνωούνται οτι θα ακολουθηθούν· το αντίθετο για βρόχους προς τα εμπρός.
- ιστορικό ($profile$) εκτέλεσης προγράμματος.
- το πρόγραμμα εκτελείται και οι βρόχοι προγνωούνται ανάλογα με την συμπεριφορά που κατέγραψε το ιστορικό.

Ευριστική/Ιστορική Πρόγνωση Βρόχων



Συνολική Απόδοση του DLX



Εξαιρέσεις σε Κλιμακωτές Αρχιτεκτονικές

- οι εξαιρέσεις είναι ειδικές περιστάσεις όπου η ροή του προγράμματος διακόπτεται και έλεγχος μεταφέρεται σε ειδικό πρόγραμμα για τον χειρισμό τους.

- τέτοιες ειδικές περιστάσεις περιλαμβάνουν:

διακοπή συσκευής εισόδου/εξόδου, κλήση λειτουργικού, εκτέλεση 1 προς 1 (*tracing*), υπερχείλιση σε πράξεις, λάθος σελίδας μνήμης, παραβίαση προστασίας μνήμης, εκτέλεση μη-εκτελέσιμης εντολής, κ.λ.π. ...

Διαδικασία Εξαίρεσης

1. στον επόμενο κύκλο IF εισάγεται εντολή βρόχου εξαίρεσης (π.χ. int, $trap$).

2. μέχρι την εκτέλεση του βρόχου εξαίρεσης αναίρουνται οι εγγραφές για την λανθάνουσα εντολή και αυτές που ακουλουθούν.

3. ο PC της λανθάνουσας εντολής σώζεται απο το πρόγραμμα χειρισμού εξαίρεσης και εκέι θα επιστραφεί ο ελεγχος.

- Ακριβείς Εξαιρέσεις ($Precise\ Exceptions$): η λανθάνουσα εντολή σταματείται, οι προηγούμενες της ολοκληρώνται, ενώ οι επόμενες μπορούν να ξαναρχίσουν.

Εξαιρέσεις στον DLX

Πιθανές εξαιρέσεις ανα στάδιο της κλιμάκωσης:

Στάδιο	Τύπος Εξαίρεσης
IF	λάθος σελίδας μνήμης, μη-ευθυγραμισμένη διεύθυνση, παραβίαση προστασίας μνήμης
ID	μη αναγνωρίσιμος/επιτρέψιμος κώδικας εντολής
EX	αριθμητική εξαίρεση
MEM	λάθος σελίδας μνήμης, μη-ευθυγραμισμένη διεύθυνση, παραβίαση προστασίας μνήμης
WB	-

- λόγω της κλιμάκωσης οι εξαιρέσεις μπορεί να εμφανιστούν με διαφορετική σειρά (εκτός σειράς) απο των εντολών:

LW	IF	ID	EX	MEM(!)	WB	
ADD		IF(!)	ID	EX	MEM	WB

Εξαιρέσεις στον DLX

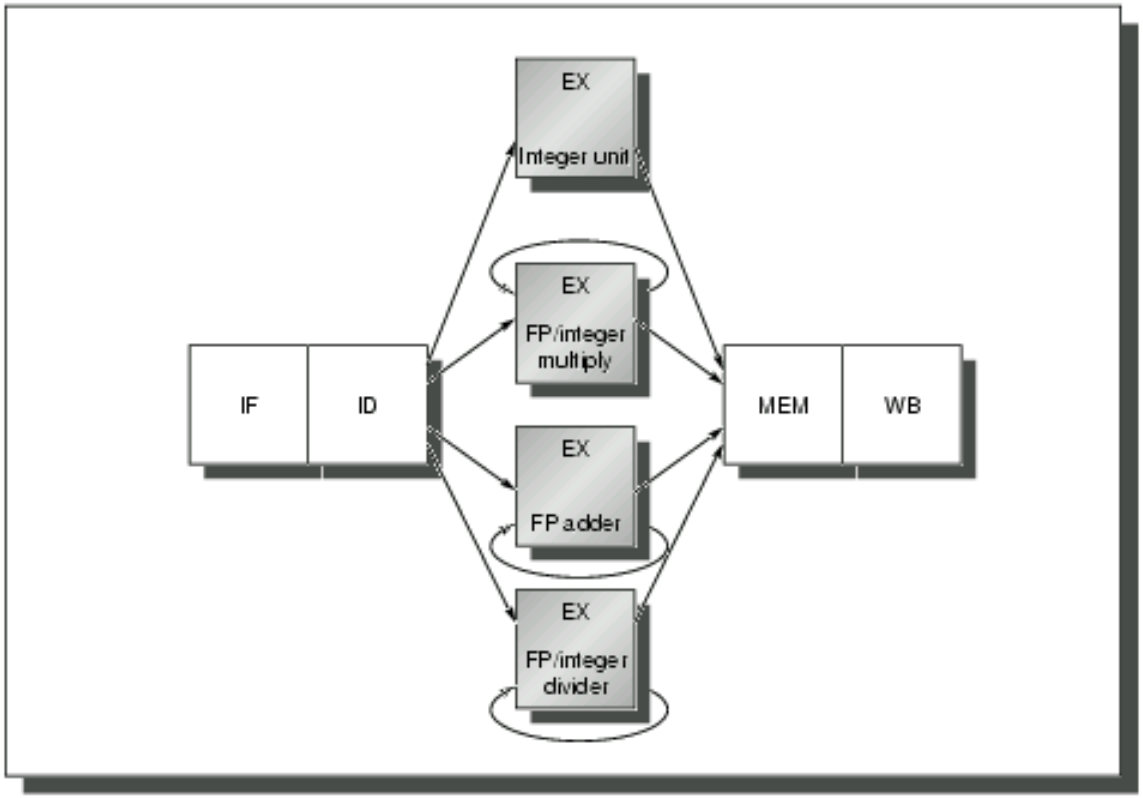
- οι εξαιρέσεις πρέπει να χειριστούν με την σειρά των εντολών (ακριβείς) και όχι με χρονική σειρά.
- έτσι οι εξαιρέσεις πρέπει να καθυστερούνται.
- αυτό μπορεί να επιτευχθεί με μία λέξη που προχωράει μαζί με την εντολή στην κλιμάκωση, το διάνυσμα κατάστασης εξαίρεσεων (***exception status vector***).
- όταν προκαλεσθεί εξαίρεση απο μια εντολή: (α) η εξαίρεση γράφεται στο διάνυσμα κατάστασης εξαίρεσεων της και (β) αναιρούνται οι πράξεις εγγραφής.
- όταν η εντολή φτάσει στο στάδιο WB οι εξαιρέσεις που έχουν γραφεί στο διάνυσμα κατάστασης εξαίρεσεων της θα χειριστούν με την σειρά της κλιμάκωσης.

Λειτουργίες Πολλαπλών Κύκλων/Πολλαπλές Μονάδες

- πολλές λειτουργίες δέν μπορούν να εκτελεστούν σε ένα κύκλο, π.χ. πράξεις ρητών (*floating-point*) αριθμών.
- έτσι ο κύκλος EX πρέπει να επαναληφθεί.
- επίσεις μπορεί να υπάρχουν πολλαπλές μονάδες εκτέλεσης, για ρητούς καί ακεραίους αριθμούς.
- μπορούμε να επεκτήνουμε π.χ. τον DLX έτσι ώστε να περιλαμβάνει τέσσερις λειτουργικές μονάδες:

1. κεντρική ακέραια μονάδα για βρόχους, εντολές μνήμης και ακέραιες πράξεις.
2. πολλαπλαστής ακεραίων και ρητών.
3. ρητός προσθετής που χειρίζεται πρόσθεση/αφαίρεση και μετατροπή.
4. ρητός και ακέραιος διαιρέτης.

Ο DLX με τέσσερις λειτουργικές μονάδες



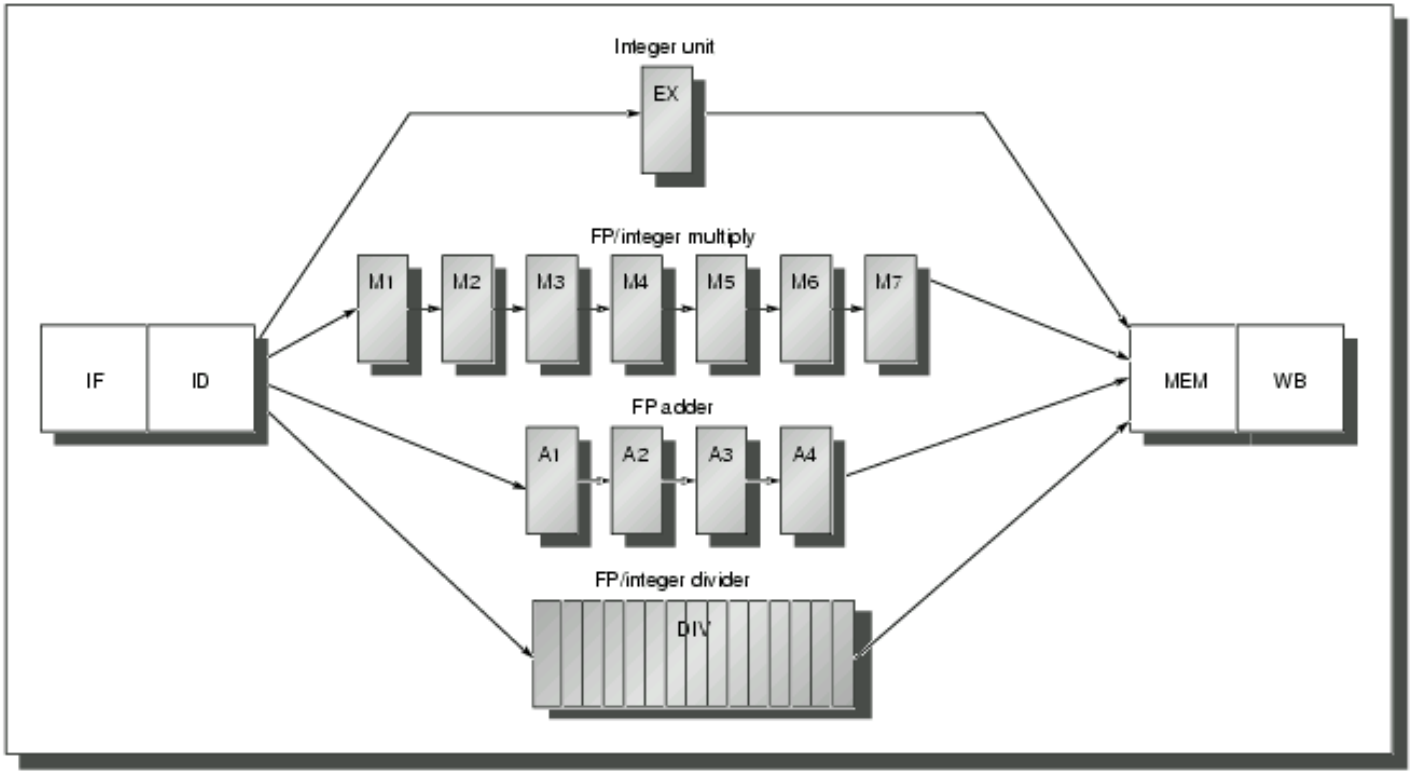
Ο DLX με τέσσερις λειτουργικές μονάδες

- οι λειτουργικές μονάδες χαρακτηρίζονται απο δύο παράγοντες: καθυστέρηση ($latency$) καί διάστημα εκκίνησης ($initiation\ interval$).
- καθυστέρηση: ο αριθμός κύκλων ανάμεσα στην εντολή που παράγει ένα αποτέλεσμα και την εντολή που το χρησιμοποιεί.
- διάστημα εκκίνησης: ο αριθμός κύκλων ανάμεσα σε εντολές του ιδίου τύπου.

Παράδειγμα τιμών των λειτουργικών μονάδων του DLX:

Λειτουργική μονάδα	Καθυστέρηση	Διάστημα Εκκίνησης
Κεντρική Ακέραια	0	1
Μνήμη Δεδομένων	1	1
Προσθετής Ρητών	3	1
Πολλαπλασιαστης Ρητών/Ακεραίων	6	1
Διαιρετής Ρητών/Ακεραίων	24	25

Ο DLX με κλιμακωτές λειτουργικές μονάδες



Χρονισμός Εντολών πολλαπλών κύκλων

MULTD	IF	ID	$M1$	$M2$	$M3$	$M4$	$M5$	$M6$	**M7**	MEM	WB
ADDD		IF	ID	$A1$	$A2$	$A3$	**A4**	MEM	WB		
LD			IF	ID	EX	**MEM**	WB				
SD				IF	ID	EX	MEM	WB			

- η νέα κλιμάκωση έχει πολλαπλές λειτουργικές μονάδες και υλοποιεί λειτουργιές που απαιτούν πολλαπλούς κύκλους.

- οι συνθήκες ύπαρξης κινδύνων δεδομένων και η υλοποίηση προώθησης αλλάζουν.

Κίνδυνοι σε Κλιμακώσεις μεταβλητών καθυστερήσεων

- Δομικοί κίνδυνοι είναι πιθανοί σε μονάδες όπως ο διαιρέτης. Αυτοί πρέπει να ανιχνευτούν και η ροή εντολών να σταματήσει.

- Λόγω των μεταβλητών κύκλων που διαρκούν οι εντολές ο αριθμός των εγγραφών των καταχωρητών μπορεί να είναι μεγαλύτερος απο 1.

- Κίνδυνοι WAW είναι δυνατοί μιά και οι εντολές δεν φτάνουν στο στάδιο WB στην αρχική σειρά. WAR δεν είναι δυνατοί γιατί η ανάγνωση γίνεται στο ID.

Κίνδυνοι σε Κλιμακώσεις μεταβλητών καθυστερήσεων

- Οι εντολές ολοκληρώνονται με διαφορετική σειρά απο ότι στέλνονται στις λειτουργικές μονάδες δημιουργώντας προβλήματα στις εξαιρέσεις.

- Λόγω του οτι πολλές λειτουργίες έχουν μεγάλες καθυστερήσεις σταματήματα RAW θα είναι πιο συχνά.

Κίνδυνοι σε Κλιμακώσεις μεταβλητών καθυστερήσεων

Παράδειγμα καθυστέρησεων RAW:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
LD F4, 0(R2)	IF	ID	EX	MEM	WB												
MULTD F0, F4, F6		IF	ID	st	$M1$	$M2$	$M3$	$M4$	$M5$	$M6$	$M7$	MEM	WB				
ADDD F2, F0, F8			IF	st	ID	st	st	st	st	st	st	$A1$	$A2$	$A3$	$A4$	MEM	
SD 0(R2), F2					IF	st	st	st	st	st	st	ID	EX	st	st	st	MEM

Παράδειγμα σύγκρουσης στο στάδιο WB:

	1	2	3	4	5	6	7	8	9	10	11
MULTD F0, F4, F6	IF	ID	$M1$	$M2$	$M3$	$M4$	$M5$	$M6$	$M7$	MEM	WB
...		IF	ID	EX	MEM	WB					
...			IF	ID	EX	MEM	WB				
ADDD F2, F4, F6				IF	ID	$A1$	$A2$	$A3$	$A4$	MEM	WB
...					IF	ID	EX	MEM	WB		
...						IF	ID	EX	MEM	WB	
LD F2, 0(R2)							IF	ID	EX	MEM	WB

Υλοποίηση ελέγχου κινδύνων σε Κλιμακώσεις μεταβλητών καθυστερήσεων

Οι παρακάτω έλεγχοι είναι αναγκαίοι στο στάδιο ID:

- Έλεγχος για δομικούς κινδύνους - αναμονή μέχρι και η απαραίτητη λειτουργική μονάδα να είναι διαθέσιμη αλλα και η θύρα εγγραφής όταν χρειαστεί.

- Έλεγχος για κινδύνους RAW - έλεγχος για το άν οι πηγαίοι της παρούσας είναι το αποτέλεσμα εντολών που βρίσκονται στα στάδια εκτέλεσης, π.χ. $ID/A1$, $A1/A2$ ...

- Έλεγχος για κινδύνους WAW - έλεγχος για το άν κάποια εντολή στα στάδια $A1$, ..., $A4$, DIV, $M1$, ..., $M7$ έχει το ίδιο αποτέλεσμα με την παρούσα. Αν ναί η παρούσα πρέπει να περιμένει.

Εξαιρέσεις σε Κλιμακώσεις μεταβλητών καθυστερήσεων

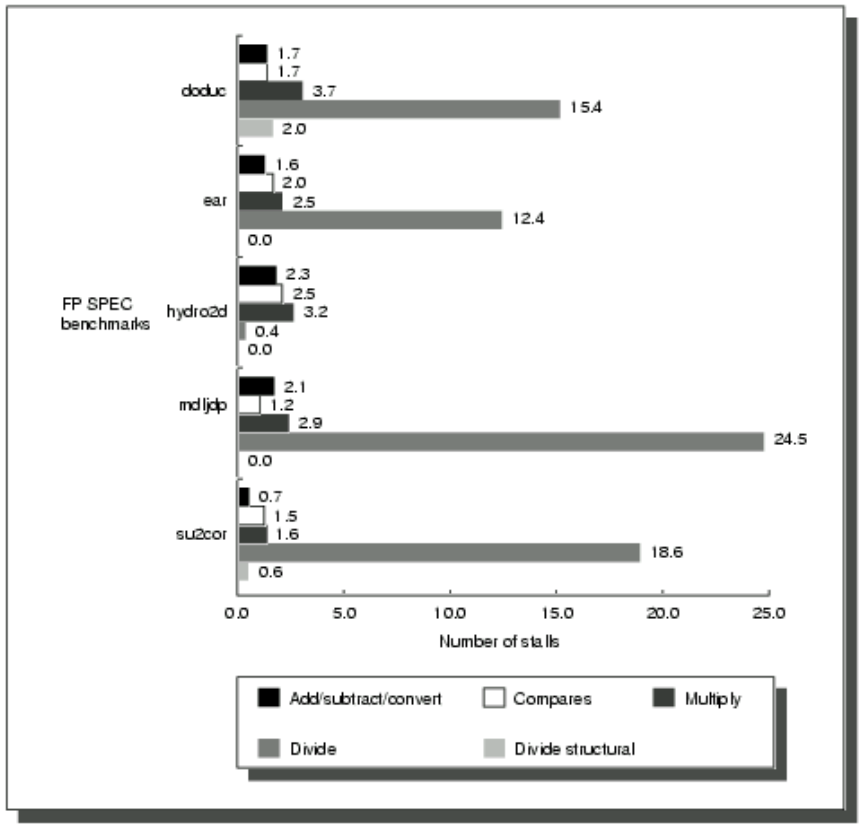
Οι μεταβλητές καθυστερήσεις δυσκολεύουν την υλοποίσης Ακριβών Εξαιρέσεων, π.χ. :

```
DIVF F0, F2, F4
ADDF F10, F10, F8
SUBF F12, F12, F14
```

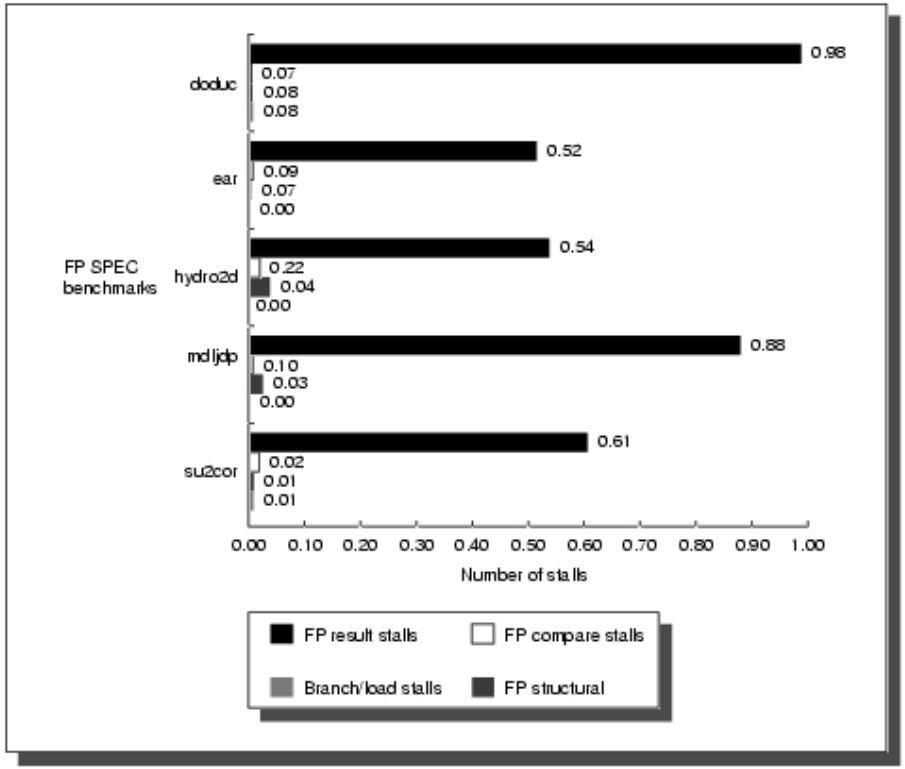
- η $ADDF$ μπορεί να ολοκληρωθεί, η $DIFV$ όχι και η $SUBF$ να προκαλέσει εξαίρεση!

- ή η $ADDF$ μπορεί να ολοκληρωθεί και η $DIVF$ να προκαλέσει εξαίρεση!

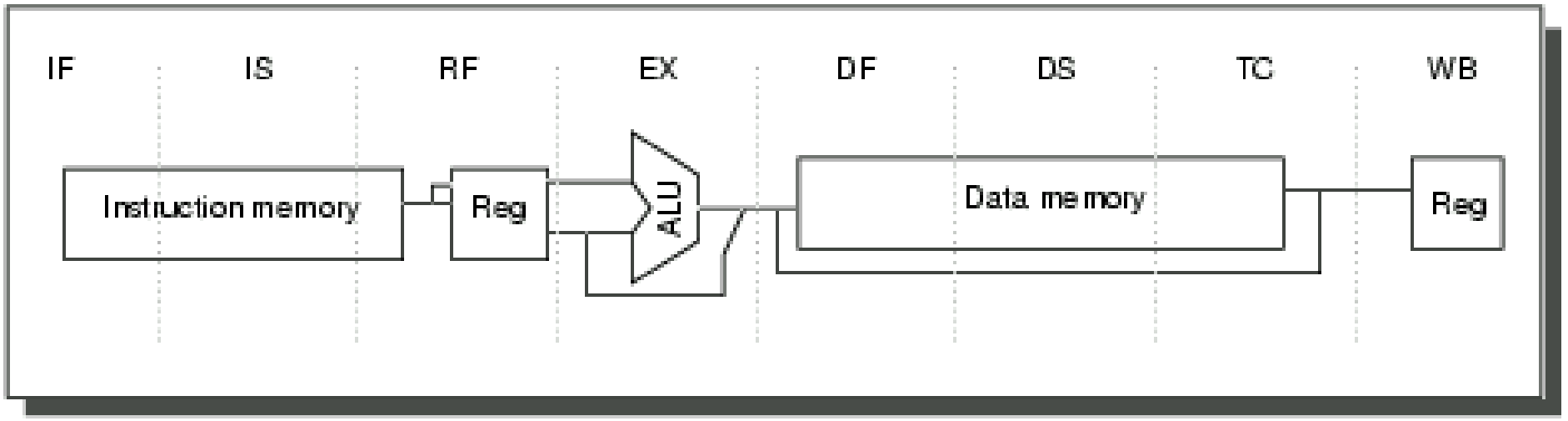
Καθυστερήσεις ανα λειτουργία στην κλιμάκωση ρητών του DLX



Καθυστερήσεις στην κλιμάκωση ρητών του DLX



Παράδειγμα: Η κλιμάκωση του $MIPS\ R4000$



Τα στάδια του $MIPS\ R4000$

Η λειτουργία του κάθε σταδίου του $MIPS\ R4000$ είναι:

- IF - πρώτο μισό της πρόσβασης εντολής· διαλέγεται ο PC και ξεκινά η πρόσβαση του $cache$.
- IS - δεύτερο μισό της πρόσβασης εντολής, ολοκληρώνει την πρόσβαση του $cache$.
- RF - αποκωδικωποίηση εντολής και ανάγνωση δρώμενων, έλεγχος κινδύνων και πρόσβασης του $cache$.
- EX - εκτέλεση· περιλαμβάνει υπολογισμό διεύθυνσης μνήμης, λειτουργία ALU, υπολογισμό διεύθυνσης και αξιολόγηση βρόχου.
- DF - πρώτο μισό της πρόσβασης $cache$ μνήμης.
- DS - δεύτερο μισό της πρόσβασης εντολής, ολοκληρώνει την πρόσβαση της μνήμης.
- TC - ελέγχος απάντησης μνήμης (hit ή $miss$).
- WB - εγγραφή δεδομένων για μνήμη και καταχωρητές.

Καθυστέρηση $loads$ στον $MIPS$ $R4000$

	1	2	3	4	5	6	7	8	9
LW R1, ...	IF	IS	RF	EX	DF	DS	TC	WB	
ADD R2, R1, ...		IF	IS	RF	stall	stall	EX	DF	DS
SUB R3, R1, ...			IF	IS	stall	stall	RF	EX	DF
OR R4, R1, ...				IF	stall	stall	IS	RF	EX

- Η προώθηση απαιτεί αναμονή 2 κύκλων.

Καθυστέρηση βρόχων στον $MIPS\ R4000$

Βρόχος που ακολουθείται:

	1	2	3	4	5	6	7	8	9
Βρόχος	IF	IS	RF	EX	DF	DS	TC	WB	
Καθυστέρηση		IF	IS	RF	EX	DF	DS	TC	WB
Κενή			κενό	κενό	κενό	κενό	κενό	κενό	κενό
Κενή				κενό	κενό	κενό	κενό	κενό	κενό
Προορισμός					IF	IS	RF	EX	DF

Βρόχος που δεν ακολουθείται:

	1	2	3	4	5	6	7	8	9
Βρόχος	IF	IS	RF	EX	DF	DS	TC	WB	
Καθυστέρηση		IF	IS	RF	EX	DF	DS	TC	WB
Βρόχος + 2			IF	IS	RF	EX	DF	DS	TC
Βρόχος + 3				IF	IS	RF	EX	DF	DS

Η κλιμάκωση ρητών του $MIPS\ R4000$

- η κλιμάκωση ρητών έχει 3 Λ.Μ.: ένα διαιρετή, ένα πολλαπλασιαστή και ένα προσθετή.
- λειτουργίες ρητών διαρκούν απο 2 μέχρι 112 κύκλους (ρίζα).
- η κλιμάκωση έχει 8 εσωτερικά στάδια που χρησιμοποιούνται ανάλογα με την λειτουργία.

Η κλιμάκωση ρητών του $MIPS\ R4000$

Στάδια:

A	Προσθετής	Πρόσθεση βάσης
D	Διαιρετής	Στάδιο διαίρεσης
E	Πολλαπλασιαστής	Στάδιο ελέγχου εξαίρεσης
M	Πολλαπλασιαστής	Πρώτο στάδιο πολλαπλασιασμού
N	Πολλαπλασιαστής	Δεύτερο στάδιο πολλαπλασιασμού
R	Προσθετής	Προσέγγιση
S	Προσθετής	Μετατόπιση
U		Αποσυμπίεση

Η κλιμάκωση ρητών του $MIPS\ R4000$

Αντιστοιχίες λειτουργιών και σταδίων, καθυστερήσεις και διαστήματα εκκίνησης:

$[S + A =$ κύκλος που και τα δύο στάδια S και A χρησιμοποιούνται$]$

Λειτουργία	Καθυστέρηση	Δ. Εκκίνησης	Στάδια
Πρόσθεση/Αφ.	4	3	U, $S+A$, $A+R$, $R+S$
Πολλαπλασιασμός	8	4	U, $E+M$, M, M, M, N, $N+A$, R
Διαίρεση	36	35	U, A, R, D^{27}, $D+A$, $D+R$, $D+A$, $D+R$, A, R
Ρίζα	112	111	U, E, $(A+R)^{108}$, A, R
Άρνηση	2	1	U, S
Απ. Τιμή	2	1	U, S
Σύγκριση	3	2	U, A, R

Εκτέλεση στην κλιμάκωση ρητών του $MIPS\ R4000$

- Πολλαπλασιασμός που ακολουθείται απο πρόσθεση απο 1 εώς 7 κύκλους:

Operation	Iss./St.	0	1	2	3	4	5	6	7	8	9
MUL	Iss.	U	M	M	M	M	N	$N+A$	R		
ADD	Iss.		U	$S+A$	$A+R$	$R+S$					
	Iss.			U	$S+A$	$A+R$	$R+S$				
	Iss.				U	$S+A$	$A+R$	$R+S$			
	St.					U	$S+A$	**A+R**	**R+S**		
	St.						U	**S+A**	**A+R**	$R+S$	
	Iss.							U	$S+A$	$A+R$	$R+S$
	Iss.								U	$S+A$	$A+R$

Εκτέλεση στην κλιμάκωση ρητών του $MIPS\ R4000$

- Πρόσθεση που ακολουθείται απο πολλαπλασιασμό:

Operation	Iss./St.	0	1	2	3	4	5	6	7	8	9
ADD	Iss.	U	$S+A$	$A+R$	$R+S$						
MUL	Iss.		U	M	M	M	M	N	$N+A$	R	
	Iss.			U	M	M	M	M	N	$N+A$	R

Εκτέλεση στην κλιμάκωση ρητών του $MIPS\ R4000$

- η διαίρεση μπορεί να σταματήσει μια πρόσθεση που αρχίζει στο τέλος της.

Operation	Iss./St.	25	26	27	28	29	30	31	32	33	34
DIV	Iss.	D	D	D	D	D	$D+A$	$D+R$	$D+A$	$D+R$	A
ADD	Iss.		U	$S+A$	$A+R$	$R+S$					
	Iss.			U	$S+A$	$A+R$	$R+S$				
	St.				U	$S+A$	**A+R**	**R+S**			
	St.					U	**S+A**	**A+R**	$R+S$		
	St.						U	$S+A$	**A+R**	**R+S**	
	St.							U	**S+A**	**A+R**	$R+S$
	St.								U	$S+A$	**A+R**
	St.									U	$S+A$
	Iss.										U

Εκτέλεση στην κλιμάκωση ρητών του $MIPS\ R4000$

- διαίρεση μετά απο πρόσθεση:

Operation	Iss./St.	1	2	3	4	5	6	7	8	9
ADD	Iss.	U	$S+A$	$A+R$	$R+S$					
DIV	St.		U	**A**	**R**	D	D	D	D	D
	Iss.			U	A	R	D	D	D	D
	Iss.				U	A	R	D	D	D

Απόδοση της κλιμάκωσης του $MIPS\ R4000$

Τέσσερις λόγοι καθυστερήσεων:

1. καθυστερήσεις λόγω ***loads*** - άν το αποτέλεσμα του *load* χρησιμοποιείται 1 ή 2 κύκλους μετά.

2. καθυστερήσεις βρόχων - δύο κύκλοι καθυστέρηση σε κάθε βρόχο που ακολουθήται.

3. καθυστερήσεις αποτελεσμάτων - λόγω RAW κινδύνων στα δρώμενα.

4. δομικές καθυστερήσεις - συγκρούσεις στην χρήση των μονάδων.