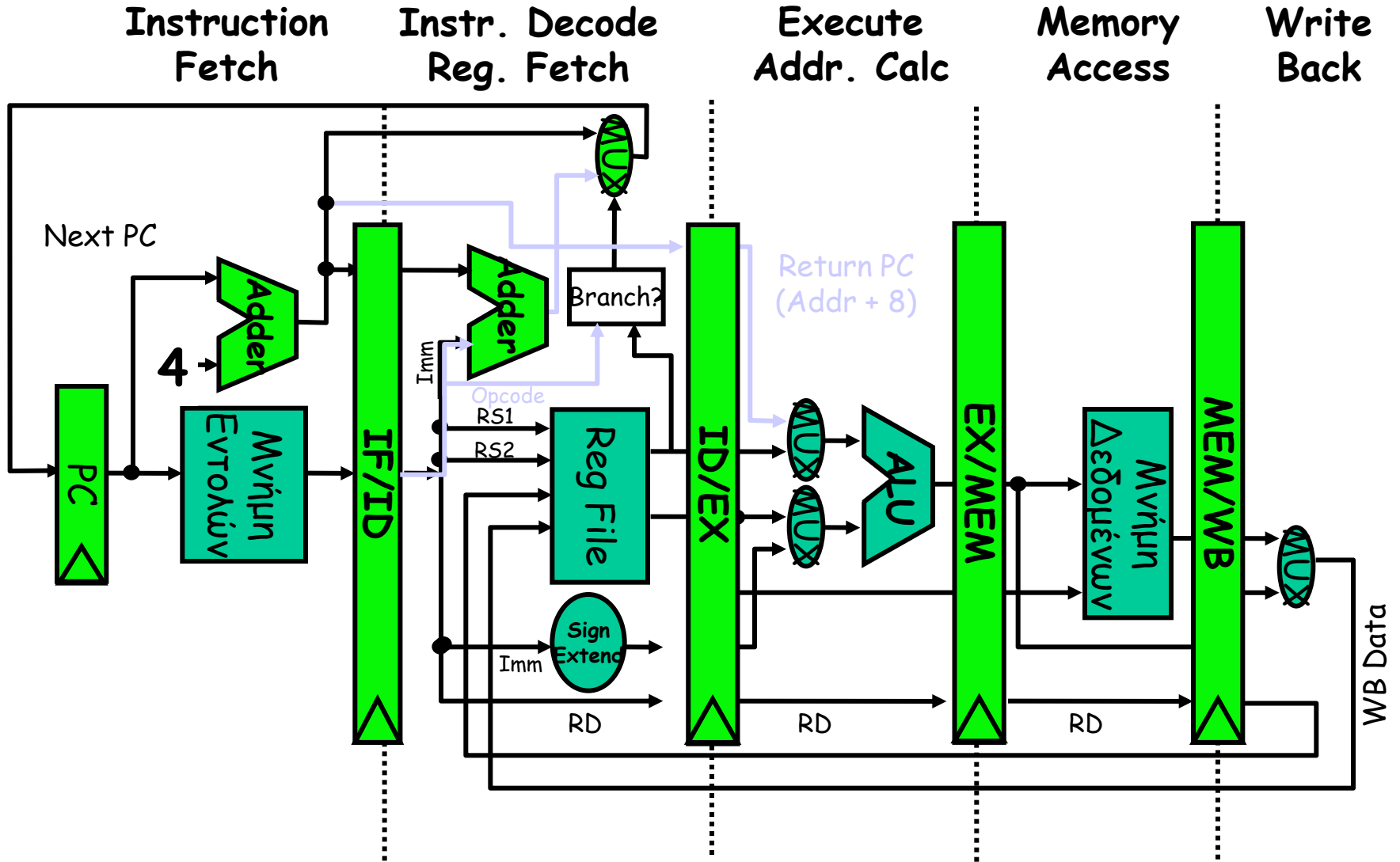


HY425
Αρχιτεκτονική Υπολογιστών

Προχωρημένες Τεχνικές Pipelining.

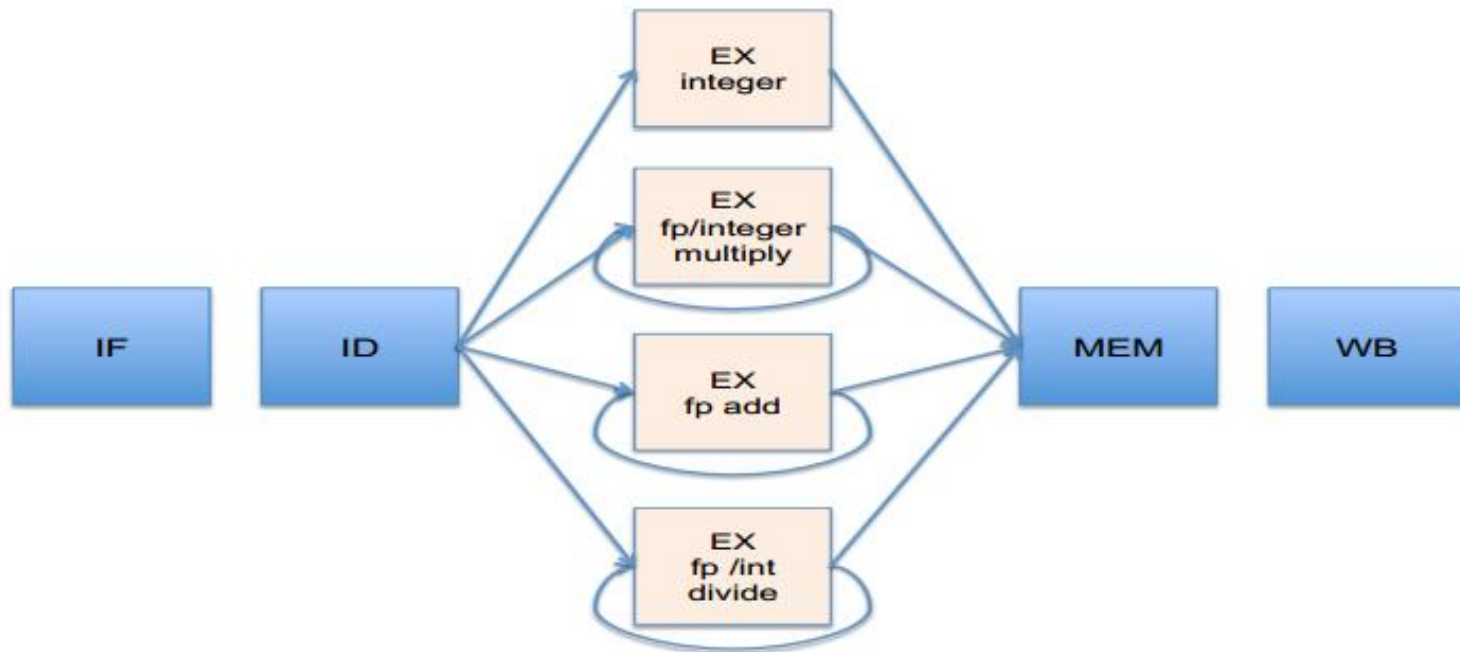
Ιάκωβος Μαυροειδής

Επεξεργαστής DLX



Pipelines with variable instruction latencies

Loops in the EX stage

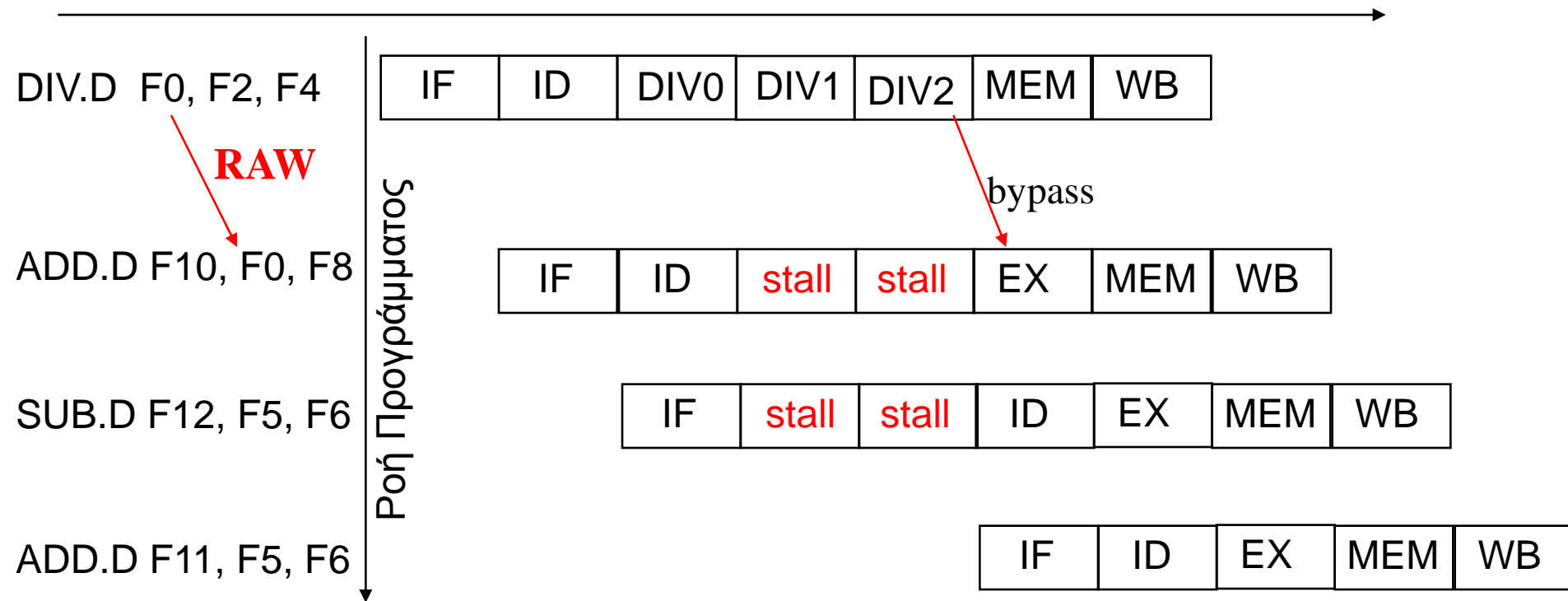


FP instructions need to have multiple execution cycles.
DLX pipeline does not work efficiently.

Εκτέλεση Εντολών σε σειρά in order execution.

(Γνωστό από προηγούμενα μαθήματα)

Χρόνος



Αν η διαίρεση χρειάζεται 10 κύκλους DIV0, DIV1 ... DIV9 ;
9 stalls και 9 χαμένοι κύκλοι...

In Order vs Out Of Order (O3)

Execution of a stage **in order** = The instructions enter the stage in programming order

Execution of a stage **out of order** = No constraint regarding the order the instructions enter the stage

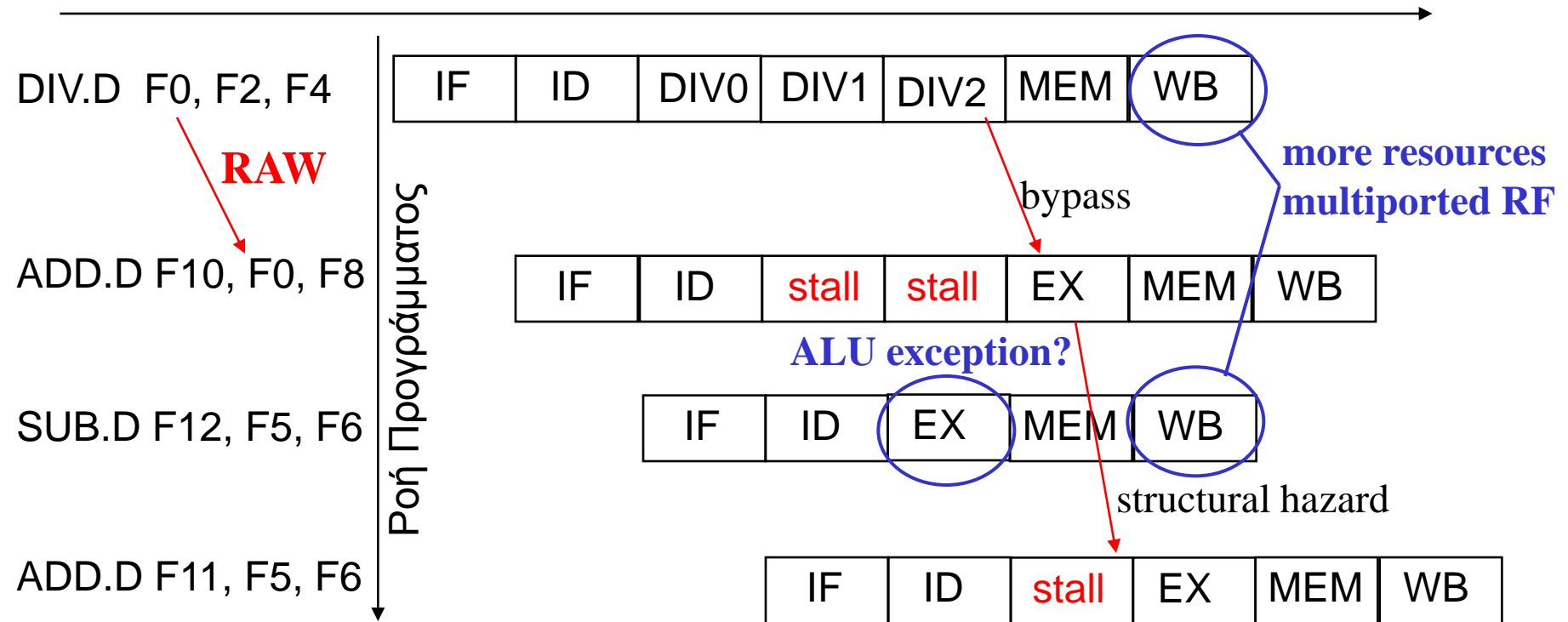
Example:

- in order fetch
- in order decode
- out of order execution
- out of order write back (commit)

Εκτέλεση Εντολών εκτός σειρά out of order execution

(Το θέμα που θα μας απασχολήσει στα επόμενα μαθήματα)

Χρόνος



Οι εντολές στέλνονται για εκτέλεση μόλις είναι διαθέσιμες οι τιμές των source registers και εφόσον δεν υπάρχει structural hazard.

Ακόμα και αν η διαίρεση χρειάζεται 10 κύκλους η pipeline κολλά για 1 κύκλο (1 stall).

Instruction Level Parallelism

Instruction Level Parallelism (ILP)

- Potential overlap among instructions
- Παράλληλη ή out-of-order εκτέλεση εντολών.
- Επέκταση της απλής pipelining αρχιτεκτονικής

Loop Level Parallelism

- Εκμετάλλευση ILP μεταξύ εντολών από διαφορετικά βασικά blocks.
e.g. Iterations of a loop
for (i=1; i<=100; i++) x[i] = x[i] + y[i];

Προσοχή να διατηρούνται

1. Data flow

Η πραγματική ροή των τιμών ανάμεσα στις εντολές που δημιουργούν τα αποτελέσματα και τις εντολές που τα χρησιμοποιούν. (RAW, WAW, WAR)

2. Exception Behavior

Αλλαγή στην σειρά εκτέλεσης των εντολών να μην αλλάζει την σειρά των που δημιουργούνται τα exceptions ή να μην δημιουργούνται καινούρια exceptions.
(next slide)


Exception behavior και Data flow

Program Order: Πρέπει το αποτέλεσμα της εκτέλεσης των εντολών να είναι το ίδιο με την σειριακή εκτέλεση τους.

Διατήρηση του Data Flow: Διατήρηση των data dependences και των control dependences

Exception behavior: Η αλλαγή της σειράς εκτέλεσης των εντολών δεν πρέπει να δημιουργήσει νέα exceptions.

DADDU	R2, R3, R4
BEQZ	R2, L1 <i>εκτέλεση εντολής πριν την branch?</i>
LW	R1, 0(R2)



L1:

Τεχνικές ελάττωσης stalls.

$CPI = \text{Ideal CPI} + \text{Structural stalls per instruction} + \text{RAW stalls per instruction} + \text{WAR stalls per instruction} + \text{WAW stalls per instruction} + \text{Control stalls per instruction}$

Στο 1ο παράδειγμα (in order execution) είχαμε 2 RAW stalls και κανένα άλλο stall.

Στο 2ο παράδειγμα (out order execution) είχαμε 1 Structural stall και κανένα άλλο stall.

Θα μελετήσουμε δύο ειδών τεχνικές

Δυναμικές δρομολόγηση εντολών (hardware)

- Scoreboard (ελάττωση RAW stalls)
- Register Renaming
 - α) Tomasulo
 - β) Reorder Buffer(ελάττωση WAR και WAW stalls)
- Branch prediction (ελάττωση Control stalls)

Στατικές (software/compiler)

- Loop Unrolling
- Software Pipelining
- Trace Scheduling

Μπορούμε να χρησιμοποιήσουμε το Hardware για να έχουμε CPI κοντά στο 1;

- Γιατί στο Hardware κατά την εκτέλεση του προγράμματος;
 - Δουλεύει όταν δεν ξέρουμε τις πραγματικές εξαρτήσεις κατά το compile (π.χ. memory references).
 - Η δουλειά του compiler είναι πιο απλή. Διαφορετικά πρέπει να γνωρίζει το συγκεκριμένο Hardware.
 - Control dependences
 - Ο assembly κώδικας μιας μηχανής τρέχει καλά και σε άλλη.
- Ιδέα κλειδί : Επέτρεψε εντολές μετά το stall να εκτελούνται (κάνε stall μόνο την εντολή που είναι data dependent)

```
DIV.D F0, F2, F4
```

```
ADD.D F10, F0, F8
```

```
SUB.D F12, F8, F14
```

- **Out-of-order** execution => **out-of-order** completion.

Εξαρτήσεις Μεταξύ Εντολών (Depedences)

- Ποιές είναι οι πηγές των stalls/bubbles; Εντολές που χρησιμοποιούν ίδιους registers.

- *Παράλληλες* εντολές μπορούν να εκτελεστούν διαδοχικά χωρίς να δημιουργούν stalls (αγνοώντας structural hazards). Π.χ.

DIV.D F0, F2, F4

ADD.D F10, F1, F3

- *Εξαρτήσεις* μεταξύ εντολών μπορούν να οδηγήσουν σε stalls. Π.χ.

DIV.D F0, F2, F4

**RAW**

ADD.D F10, F0, F3

- Οι εξαρτήσεις μεταξύ εντολών περιορίζουν την σειρά εκτέλεσης των εντολών (in order execution) π.χ. η ADDD πρέπει να εκτελεστεί μετά την DIVD στο 2ο παράδειγμα. Ενώ παράλληλες εντολές μπορούν να εκτελεστούν ανάποδα (out of order execution) π.χ. η ADDD μπορεί να εκτελεστεί πριν την DIVD στο 1ο παράδειγμα.

Εξαρτήσεις Μεταξύ Εντολών

• **(True) Data Dependences** : Δύο εντολές είναι data dependent όταν υπάρχει μία αλυσίδα από RAW hazards μεταξύ τους.

Loop:

```
L.D      F0, 0(R1)
          |
ADD.D    F4, F0, F2
          |
S.D      F4, 0(R1)

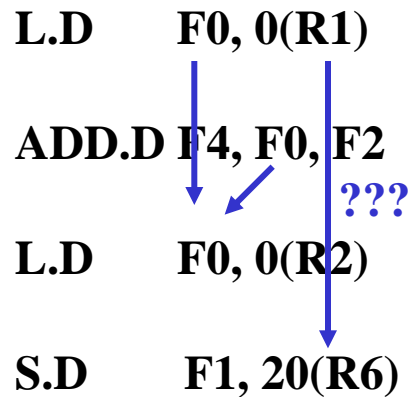
DADDUI   R1, R1, -8
          |
BNE      R1, R2, Loop
```

Τι εκτελεί το παραπάνω πρόγραμμα;

```
S.D      F0, 100(R4)
          |
          | ???
L.D      F1, 20(R6)
```

Εξαρτήσεις Μεταξύ Εντολών

• **Name Dependences** : Δύο εντολές είναι name dependent όταν υπάρχει ένα WAR (antidependence) ή WAW (output dependence) hazard μεταξύ τους.



Ποιο είναι το antidependence;

• **Control Dependences** : Εντολές εξαρτούμενες από branch εντολή.
if p1 { S1; }

Προβλήματα με Hazards.

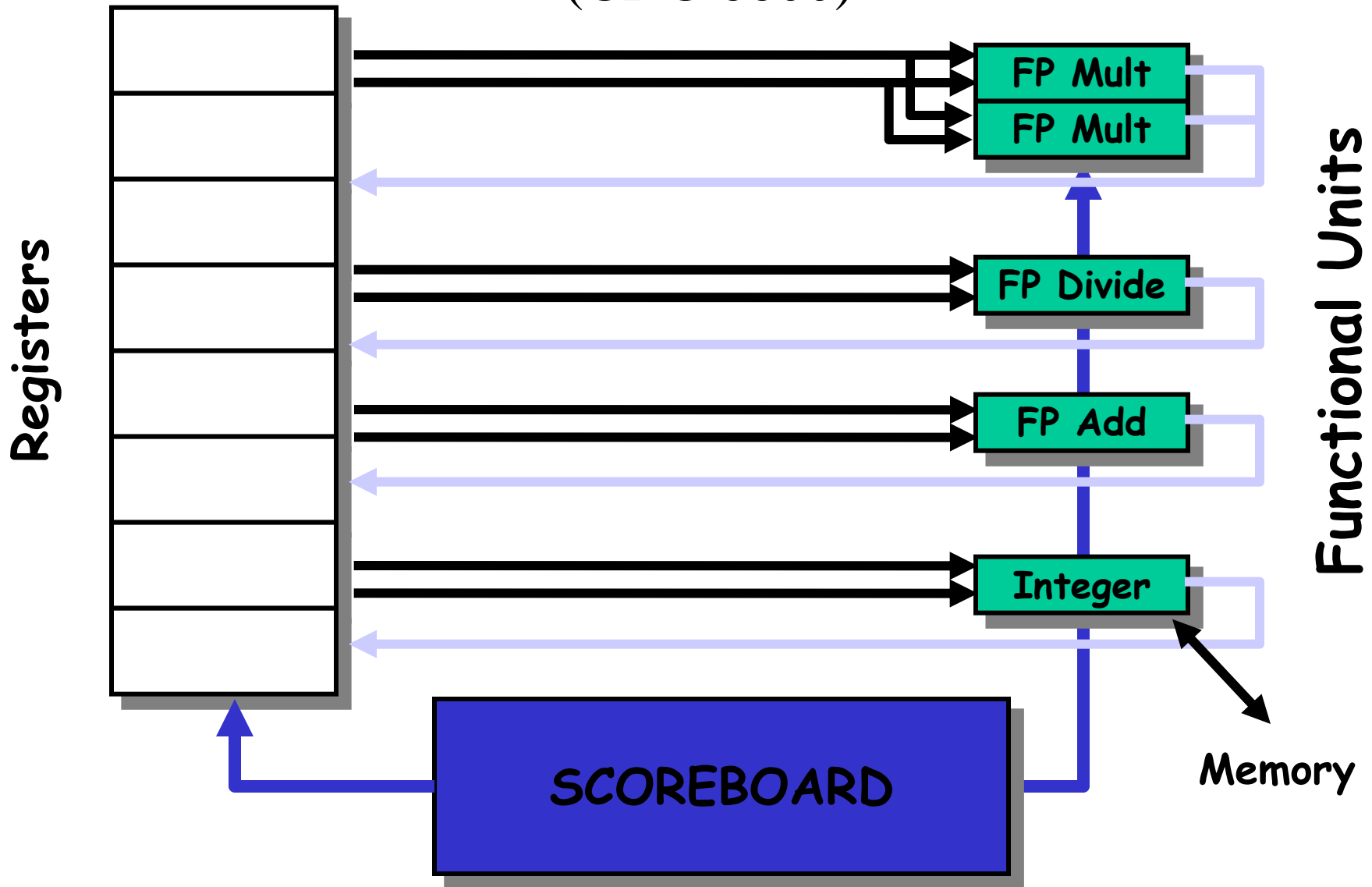
- Πώς εμποδίζουμε WAR και WAW hazards;
- Τι κάνουμε για τις μεταβλητές καθυστερήσεις των μονάδων εκτέλεσης;
 - Forwarding των RAW hazards δυσκολότερη.

Instruction	Clock Cycle Number																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
LD F6,34(R2)	IF	ID	EX	MEM	WB												
LD F2,45(R3)		IF	ID	EX	MEM	WB											
MULTD F0,F2,F4			IF	ID	stall	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	MEM	WB
SUBD F8,F6,F2				IF	ID	A1	A2	MEM	WB								
DIVD F10,F0,F6					IF	ID	stall	stall	stall	stall	stall	stall	stall	stall	stall	D1	D2
ADDD F6,F8,F2						IF	ID	A1	A2	MEM	WB						

Scoreboard : out of order execution

- Χωρισμός του ID stage στα δύο:
 1. **Έκδοση/Issue**—Αποκοδικοποίηση εντολής, έλεγχος για structural hazards. In order instruction issue.
 2. **Ανάγνωση/read operands**—Αναμονή μέχρι να μην υπάρχει data hazards, μετά διάβασε source registers. Out of order execution.
- Οι πρώτες μηχανές Scoreboards εμφανίστηκαν με το CDC6600 το 1963
- Εντολές εκτελούνται όποτε δεν είναι εξαρτόμενες από προηγούμενες εντολές (που δεν έχουν εκτελεστεί) και δεν υπάρχουν hazards.
- CDC6600: In order issue, out-of-order execution, out-of-order commit (or completion)
 - 16 functional units: 4 floating-point units, 5 units for memory references, and 7 units for integer operations
 - Δεν υπάρχει forwarding!

Αρχιτεκτονική Scoreboard (CDC 6600)



Scoreboard Implications

- Out-of-order completion => WAR, WAW hazards?
- Λύσεις για WAR:
 - Stall writeback στάδιο μέχρι οι registers να έχουν διαβαστεί
 - Διάβασε registers μόνο κατά το Read Operands στάδιο
- Λύσεις για WAW:
 - Ανίχνευσε hazard και stall issue στάδιο καινούριας εντολής μέχρι η άλλη εντολή να εκτελεστεί.
- Δεν υπάρχει register renaming!
- Πολλές εντολές στο execution στάδιο => multiple execution units or pipelined execution units
- Το Scoreboard κρατάει πληροφορίες για τις εξαρτήσεις μεταξύ των εντολών που έχουν γίνει issued.
- Scoreboard αντικαθιστά ID στάδιο με 2 στάδια: Issue και Read Operands

Four Stages of Scoreboard Control

- **Issue**—decode εντολή & record data dependences & έλεγξε για structural hazards (ID1)
 - Instructions issued in program order (for hazard checking)
 - Don't issue if **structural hazard**
 - Don't issue if instruction is **output dependent** on any previously issued but uncompleted instruction (no WAW hazards)
- **Read operands**—περίμενε μέχρι να μην υπάρχουν data hazards, μετά διάβασε μεταβλητές (ID2)
 - All real dependencies (RAW hazards) resolved in this stage, since we wait for instructions to write back data.
 - **No forwarding of data** in this model!

Four Stages of Scoreboard Control

- **Execution**— εκτέλεση στην αριθμητική μονάδα (EX)
 - The functional unit begins execution upon receiving operands. When the result is ready, it notifies the scoreboard that it has completed execution.
- **Write result**—Τέλος Εκτέλεσης (WB)
 - Stall μέχρι να μην υπάρχει WAR hazards με προηγούμενες εντολές:

Παράδειγμα:

```
DIV.D F0, F2, F4
ADD.D F10, F0, F8
SUB.D F8, F8, F14
```

CDC 6600 scoreboard would stall SUBD until ADDD reads operands

- Instructions write their results into the register file as soon as they complete execution (assuming no WAR hazard)

Data structure: 3 μέρη του Scoreboard

- **Instruction status:**
Σε ποιο από τα 4 στάδια είναι η εντολή
- **Functional unit status:**—Δείχνει την κατάσταση του functional unit (FU). 9 πεδία για κάθε functional unit
 - Busy:** Indicates whether the unit is busy or not
 - Op:** Operation to perform in the unit (e.g., + or −)
 - Fi:** Destination register
 - Fj,Fk:** Source-register numbers
 - Qj,Qk:** Functional units producing source registers Fj, Fk
 - Rj,Rk:** Flags indicating when Fj, Fk are ready
- **Register result status**—Δείχνει ποιο functional unit θα γράψει κάθε register, αν υπάρχει κάποια. Blank when no pending instructions will write that register

Detailed Scoreboard Pipeline Control

Instruction status	Wait until	Bookkeeping
Issue	$\text{Busy (FU)} = \text{No}$ and $\text{result(D)} = 0$ (struct hazard, WAW)	$\text{Busy(FU)} \leftarrow \text{yes}; \text{Op(FU)} \leftarrow \text{op};$ $\text{Fi(FU)} \leftarrow \text{'D'}; \text{Fj(FU)} \leftarrow \text{'S1'};$ $\text{Fk(FU)} \leftarrow \text{'S2'}; \text{Qj} \leftarrow \text{Result('S1')};$ $\text{Qk} \leftarrow \text{Result('S2')}; \text{Rj} \leftarrow \text{not Qj};$ $\text{Rk} \leftarrow \text{not Qk}; \text{Result('D')} \leftarrow \text{FU};$
Read operands	Rj and Rk (RAW)	$\text{Rj} \leftarrow \text{No}; \text{Rk} \leftarrow \text{No}$
Execution complete	Functional unit done	
Write result	$\forall f((\text{Fj}(f) \neq \text{Fi(FU)})$ or $\text{Rj}(f) = \text{No}) \&$ $(\text{Fk}(f) \neq \text{Fi(FU)})$ or $\text{Rk}(f) = \text{No}))$ (WAR)	$\forall f(\text{if } \text{Qj}(f) = \text{FU} \text{ then } \text{Rj}(f) \leftarrow \text{Yes});$ $\forall f(\text{if } \text{Qk}(f) = \text{FU} \text{ then } \text{Rj}(f) \leftarrow \text{Yes});$ $\text{Result(Fi(FU))} \leftarrow 0; \text{Busy(FU)} \leftarrow \text{No}$

Πατάδειγμα Scoreboard : Κύκλος 1

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>
LD	F6	34+	R2	1		
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

Functional unit status:

Time	Name	Busy	Op	dest		S1		S2		FU		FU		Fj?		Fk?	
				Fi	Fj	Fk	Qj	Qk	Rj	Rk							
	Integer	Yes	Load	F6			R2										Yes
	Mult1	No															
	Mult2	No															
	Add	No															
	Divide	No															

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
1				Integer					

Παράδειγμα Scoreboard : Κύκλος 2

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

Functional unit status:

Time	Name	Busy	Op	dest <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	Yes	Load	F6		R2				No
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
2	FU Integer								

- Issue Δεύτερο LD; Structural Hazard.

Παράδειγμα Scoreboard : Κύκλος 3

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

Functional unit status:

Time	Name	Busy	Op	dest <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	Yes	Load	F6		R2				No
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
3	FU Integer								

- Issue MULT; In order issue.

Παράδειγμα Scoreboard : Κύκλος 4

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Read Oper</i>	<i>Exec Comp</i>	<i>Write Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3				
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status:

Time	Name	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
4	FU								

Παράδειγμα Scoreboard : Κύκλος 5

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Read Oper</i>	<i>Exec Comp</i>	<i>Write Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5			
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status:

Time	Name	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	Load	F2		R3				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
5	FU Integer								

Παράδειγμα Scoreboard : Κύκλος 6

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Read Oper</i>	<i>Exec Comp</i>	<i>Write Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6		
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest Fi</i>	<i>S1 Fj</i>	<i>S2 Fk</i>	<i>FU Qj</i>	<i>FU Qk</i>	<i>Fj? Rj</i>	<i>Fk? Rk</i>
	Integer	Yes	Load	F2	F2	R3			No	No
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	No								
	Divide	No								

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
6	Mult1	Integer							

Παράδειγμα Scoreboard : Κύκλος 7

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Read Oper</i>	<i>Exec Comp</i>	<i>Write Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2	7			
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest Fi</i>	<i>S1 Fj</i>	<i>S2 Fk</i>	<i>FU Qj</i>	<i>FU Qk</i>	<i>Fj? Rj</i>	<i>Fk? Rk</i>
	Integer	Yes	Load	F2		R3				No
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Integer	Yes	No
	Divide	No								

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
7									
<i>FU</i>	Mult1	Integer			Add				

- Διάβασμα των τιμών του multiply? RAW Hazard.

Παράδειγμα Scoreboard : Κύκλος 8α

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2	7			
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	Yes	Load	F2		R3				No
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Integer	Yes	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
8	<i>FU</i> Mult1	Integer			Add	Divide			

Παράδειγμα Scoreboard : Κύκλος 8β

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2	7			
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
8	<i>FU</i> Mult1				Add	Divide			

Παράδειγμα Scoreboard : Κύκλος 9

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3 4
LD	F2	45+	R3	5	6	7 8
MULTD	F0	F2	F4	6	9	
SUBD	F8	F6	F2	7	9	
DIVD	F10	F0	F6	8		
ADDD	F6	F8	F2			

Functional unit status:

Time	Name	Busy	Op	dest <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	No								
Χρόνος →	10 Mult1	Yes	Mult	F0	F2	F4			No	No
Εκτέλεσης	Mult2	No								
	2 Add	Yes	Sub	F8	F6	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
9	<i>FU</i> Mult1				Add	Divide			

- Διάβασμα των τιμών του MULT & SUB?
Issue ADDD? Structural hazard

Παράδειγμα Scoreboard : Κύκλος 10

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9		
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
9	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
1	Add	Yes	Sub	F8	F6	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
10	<i>FU</i> Mult1				Add	Divide			

Παράδειγμα Scoreboard : Κύκλος 11

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
8	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
0	Add	Yes	Sub	F8	F6	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
11	<i>FU</i> Mult1				Add	Divide			

Παράδειγμα Scoreboard : Κύκλος 12

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
7	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	No								
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
12	<i>FU</i> Mult1					Divide			

- Διάβασμα των τιμών του DIVD? RAW Hazard.

Παράδειγμα Scoreboard : Κύκλος 13

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13			

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
6	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
13	<i>FU</i> Mult1			Add		Divide			

Παράδειγμα Scoreboard : Κύκλος 14

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14		

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
5	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
2	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
14	<i>FU</i> Mult1			Add		Divide			

Παράδειγμα Scoreboard : Κύκλος 15

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14		

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
4	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
1	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
15	<i>FU</i> Mult1			Add		Divide			

Παράδειγμα Scoreboard : Κύκλος 16

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14	16	

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
3	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
0	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
16	<i>FU</i> Mult1			Add		Divide			

Παράδειγμα Scoreboard : Κύκλος 17

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>
LD	F6	34+	R2	1	2	3 4
LD	F2	45+	R3	5	6	7 8
MULTD	F0	F2	F4	6	9	
SUBD	F8	F6	F2	7	9	11 12
DIVD	F10	F0	F6	8		
ADDD	F6	F8	F2	13	14	16

WAR Hazard!

Functional unit status:

Time	Name	Busy	Op	dest <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	No								
2	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	MULT		NO	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
17									
<i>FU</i>	Mult1			Add		Divide			

- Γιατί να μην γραφτούν τα αποτελέσματα της ADD???

Παράδειγμα Scoreboard : Κύκλος 18

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14	16	

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
1	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
18	<i>FU</i> Mult1			Add		Divide			

Παράδειγμα Scoreboard : Κύκλος 19

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14	16	

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
0	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
19	<i>FU</i> Mult1			Add		Divide			

Παράδειγμα Scoreboard : Κύκλος 20

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14	16	

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>		<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>	
	Integer	No									
	Mult1	No									
	Mult2	No									
	Add	Yes	Add	F6	F8	F2				No	No
	Divide	Yes	Div	F10	F0	F6				Yes	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
20				Add		Divide			

Παράδειγμα Scoreboard : Κύκλος 21

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21		
ADDD	F6	F8	F2	13	14	16	

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>			<i>FU</i>		<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6			No	No

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
21				Add		Divide			

- WAR Hazard τώρα εξαφανίστηκε...

Παράδειγμα Scoreboard : Κύκλος 22

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21		
ADDD	F6	F8	F2	13	14	16	22

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>		<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>	
	Integer	No									
	Mult1	No									
	Mult2	No									
	Add	No									
39	Divide	Yes	Div	F10	F0	F6				No	No

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
22	<i>FU</i> Divide								

Μετά από μερικούς κύκλους

Παράδειγμα Scoreboard : Κύκλος 61

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21	61	
ADDD	F6	F8	F2	13	14	16	22

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>		<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>						
	Integer	No									
	Mult1	No									
	Mult2	No									
	Add	No									
0	Divide	Yes	Div	F10	F0	F6				No	No

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
61	<i>FU</i> Divide								

Παράδειγμα Scoreboard : Κύκλος 62

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21	61	62
ADDD	F6	F8	F2	13	14	16	22

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>		<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>	
	Integer	No									
	Mult1	No									
	Mult2	No									
	Add	No									
	Divide	No									

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
62	<i>FU</i>								

Παράδειγμα Scoreboard : Κύκλος 62

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Read Oper</i>	<i>Exec Comp</i>	<i>Write Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21	61	62
ADDD	F6	F8	F2	13	14	16	22

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i> <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
62	<i>FU</i>								

- In-order issue; out-of-order execute & commit

CDC 6600 Scoreboard

- Speedup 1.7 from compiler; 2.5 by hand
BUT slow memory (no cache) limits benefit
- Limitations of 6600 scoreboard:
 - No forwarding hardware
 - Limited to instructions in basic block (small *window*)
 - Small number of functional units (structural hazards), especially integer/load store units
 - Do not issue on structural hazards
 - Wait for WAR hazards
 - Prevent WAW hazards