

# Προηγμένη Αρχιτεκτονική Υπολογιστών

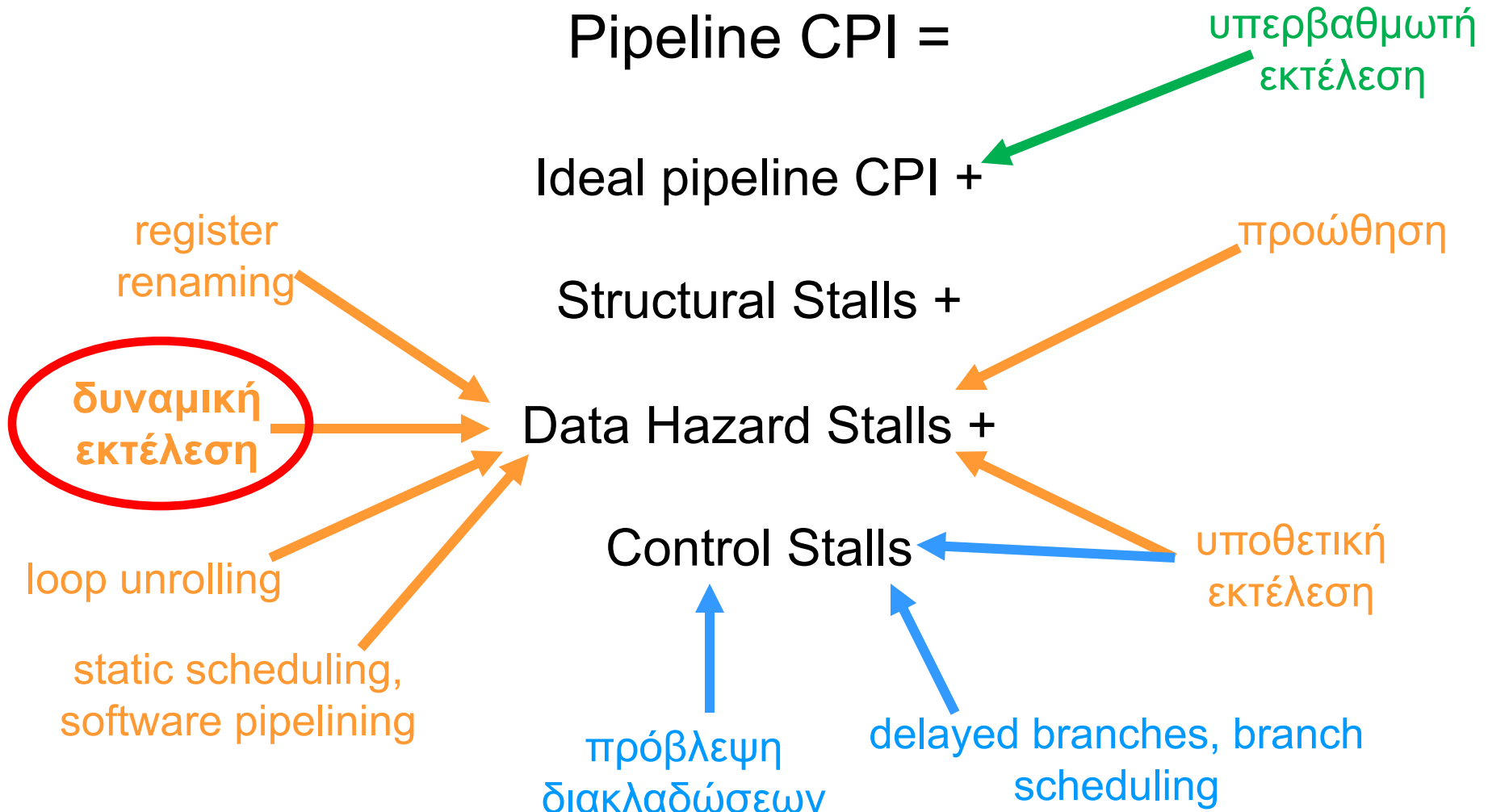
## Δυναμική Δρομολόγηση Εντολών (Dynamic Pipeline Scheduling)

Νεκτάριος Κοζύρης & Διονύσης Πνευματικάτος  
{nkoziris,pnevmati}@cslab.ece.ntua.gr

7ο εξάμηνο ΣΗΜΜΥ – Ακαδημαϊκό Έτος: 2019-20

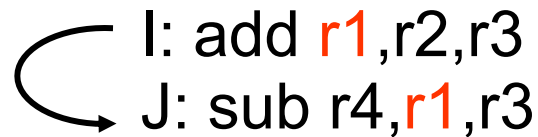
<http://www.cslab.ece.ntua.gr/courses/advcomparch/>

# Τεχνικές βελτίωσης του CPI



# Εξαρτήσεις Δεδομένων και Κίνδυνοι (Hazards)

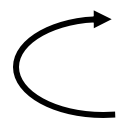
- Η J είναι **data dependent** από την I:  
Η J προσπαθεί να διαβάσει τον source operand πριν τον γράψει η I

  
I: add r1, r2, r3  
J: sub r4, r1, r3

- **Πραγματικές εξαρτήσεις (True Dependences)**
- Ροή πληροφορίας (τιμές δεδομένων)
- Προκαλούν **κινδύνους Read After Write (RAW)** στο pipeline

# Name Dependences, (1): Anti-dependences

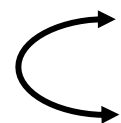
- **Name dependences:** όταν 2 εντολές χρησιμοποιούν τον ίδιο καταχωρητή ή θέση μνήμης (“name”), χωρίς όμως να υπάρχει πραγματική ροή δεδομένων μεταξύ τους
- **Anti-dependence:** η J γράφει τον r1 πριν τον διαβάσει η I

 I: sub r4,r1,r3  
J: add r1,r2,r3  
K: mul r6,r1,r7

- Προκαλούν **Write After Read (WAR) hazards** στο pipeline

## Name Dependences, (2): Output dependences

- **Output dependence:** η J γράφει τον r1 πριν τον γράψει η I

 I: sub r1,r4,r3  
J: add r1,r2,r3  
K: mul r6,r1,r7

- Προκαλούν **Write After Write (WAW)** hazards στο pipeline

# Εξαρτήσεις Δεδομένων και Hazards

- Οι εξαρτήσεις είναι ιδιότητα των **προγραμμάτων**
- Η παρουσία μιας εξάρτησης υποδηλώνει την **πιθανότητα** εμφάνισης κινδύνου, αλλά το αν θα συμβεί πραγματικά ο κίνδυνος, και το πόση καθυστέρηση θα εισάγει, είναι χαρακτηριστικό της **pipeline**
- Η σημασία των εξαρτήσεων δεδομένων
  - 1) υποδηλώνουν την πιθανότητα για hazards
  - 2) καθορίζουν τη σειρά σύμφωνα με την οποία πρέπει να υπολογιστούν τα δεδομένα
  - 3) θέτουν ένα άνω όριο στον παραλληλισμό που μπορούμε να εκμεταλλευτούμε

# ILP και Data Hazards

- Εξαιτίας των εξαρτήσεων πρέπει να διατηρούμε τη “σειρά του προγράμματος” (Program order)
- **σειρά προγράμματος:** η σειρά με την οποία θα εκτελούνταν οι εντολές, αν επεξεργάζονταν σειριακά, μία κάθε φορά, όπως υπαγορεύεται από τον πηγαίο κώδικα του προγράμματος
- σκοπός HW/SW: να εκμεταλλευτούν τον παραλληλισμό, διατηρώντας τη σειρά προγράμματος αποτρέποντας όποια αλλαγή θα μπορούσε να επηρεάσει το αποτέλεσμα του προγράμματος

# Δυναμική δρομολόγηση εντολών (1)

```
DIVD  F0,F2,F4
ADDD  F10,F0,F8
SUBD  F12,F8,F14
```

- dependence μεταξύ DIVD και ADDD
- Καμμία εξάρτηση για το SUBD. **Γιατί να περιμένει πίσω από το ADDD;**

Ισοδύναμος κώδικας

```
DIVD  F0,F2,F4
SUBD  F12,F8,F14
ADDD  F10,F0,F8
```

- Dynamic Scheduling: Αλλαγή της σειράς εκτέλεσης εντολών (**out-of-order execution**) στο υλικό κατά την εκτέλεση (runtime). Προϋποθέσεις:
  - διασφάλιση σωστής ροής δεδομένων
  - διασφάλιση σωστών εξαιρέσεων exceptions



# Δυναμική δρομολόγηση εντολών (2)

## Γιατί Δυναμική Δρομολόγηση;

- Μπορεί να διαχειριστεί περιπτώσεις όπου οι εξαρτήσεις δεν είναι γνωστές κατά το compile time (π.χ., λόγω έμμεσων αναφορών σε θέσεις μνήμης)
- Απλοποιεί τη λειτουργικότητα του compiler
- Επιτρέπει την αποδοτική εκτέλεση του προγράμματος, ανεξαρτήτως του pipeline για το οποίο μεταγλωττίστηκε και βελτιστοποιήθηκε αυτό
- Κάνει δυνατή την υποθετική εκτέλεση εντολών (εικασία, speculative execution)

## Δυναμική δρομολόγηση εντολών (3)

- Χαρακτηριστικά:
  - *in-order instruction issue*
  - *out-of-order execution*
  - *out-of-order completion*
- Η κλασική βαθμίδα ID του 5-stage pipeline χωρίζεται σε 2 «βήματα»
  - **Εκδοση (Issue)**: Αποκωδικοποίηση εντολών και έλεγχος για δομικούς κινδύνους (in order issue)
  - **Read Operands**: Διάβασμα των τελεστών όταν δεν υπάρχουν κίνδυνοι δεδομένων (οι εντολές κάνουν stall ή bypass - **εδώ εντολές μπορεί να προσπεράσουν άλλες-μπαίνουν σε ooo execution**)

# Προβλήματα στην δυναμική δρομολόγηση εντολών

- Πιθανότητα για **WAR** και **WAW** hazards

- 1.DIVD F0,F2,F4
- 2.ADDD **F6**,F0,**F8**
- 3.SUBD **F8**,F10,F14
- 4.MULD **F6**,F10,F8

- **antidependence**: (2) -> (3)
  - αν το SUBD εκτελεστεί πρώτο δημιουργείται WAR
- **output dependence**: (2) -> (4)
  - αν εκτελεστεί πρώτα το MULD δημιουργείται WAW

# Δυναμική δρομολόγηση εντολών

- **Scoreboard**

- 1963 για το CDC6600, έλεγχος με μια κεντρική δομή (scoreboard)
- Αντιμετώπιση κινδύνων WAR
  - » Stall WB μέχρι να διαβαστούν οι registers
  - » Διάβασμα καταχωρητών στο στάδιο Read Operands
- Αντιμετώπιση κινδύνων WAW
  - » Αναγνώριση κινδύνου και αποφυγή έκδοσης εντολής (stall)

- **Tomasulo's algorithm**

- 1966 για το IBM360/91
- Επίλυση WAR και WAW hazards με χρήση μετονομασίας καταχωρητών (**register renaming**)
- Πιο αποδοτική τεχνική

# Παράδειγμα Register Renaming

```
DIV.D F0,F2,F4  
ADD.D F6,F0,F8  
S.D F6,0(R1)  
SUB.D F3,F10,F14  
MUL.D F6,F10,F8
```



```
DIV.D F0,F2,F4  
ADD.D F6,F0,F8  
S.D F6,0(R1)  
SUB.D T,F10,F14  
MUL.D F6,F10,T
```

```
DIV.D F0,F2,F4  
ADD.D S,F0,F8  
S.D S,0(R1)  
SUB.D T,F10,F14  
MUL.D F6,F10,T
```




```
SUB.D T,F10,F14  
MUL.D F6,F10,T
```

```
DIV.D F0,F2,F4  
ADD.D S,F0,F8  
S.D S,0(R1)
```



# Αλγόριθμος Tomasulo

- **Reservation Stations (RS)**
  - Αποθηκεύουν τους operands των εντολών που περιμένουν να εκτελεστούν
  - Κατανεμημένα μαζί με τα **Functional Units (FUs)**
- Οι **source registers** κάθε εντολής αντικαθίστανται με το **όνομα του κατάλληλου RS**, το οποίο θα της παράσχει το απαιτούμενο input  **register renaming**
  - Αποφυγή WAR, WAW hazards
  - Περισσότερα RS από πραγματικούς registers διασφαλίζουν την αποφυγή κινδύνων εξαιτίας name dependences που δεν μπορεί να επιλύσει ένας compiler
- “προώθηση” αποτελεσμάτων από τα RS στα FU, **όχι μέσω του register file**, αλλά πάνω από το **Common Data Bus** που κάνει broadcast τα αποτελέσματα σε όλα τα FUs
- Load, Stores αντιμετωπίζονται κι αυτά ως FUs με RS

# Tomasulo for MIPS floating point + load-store unit

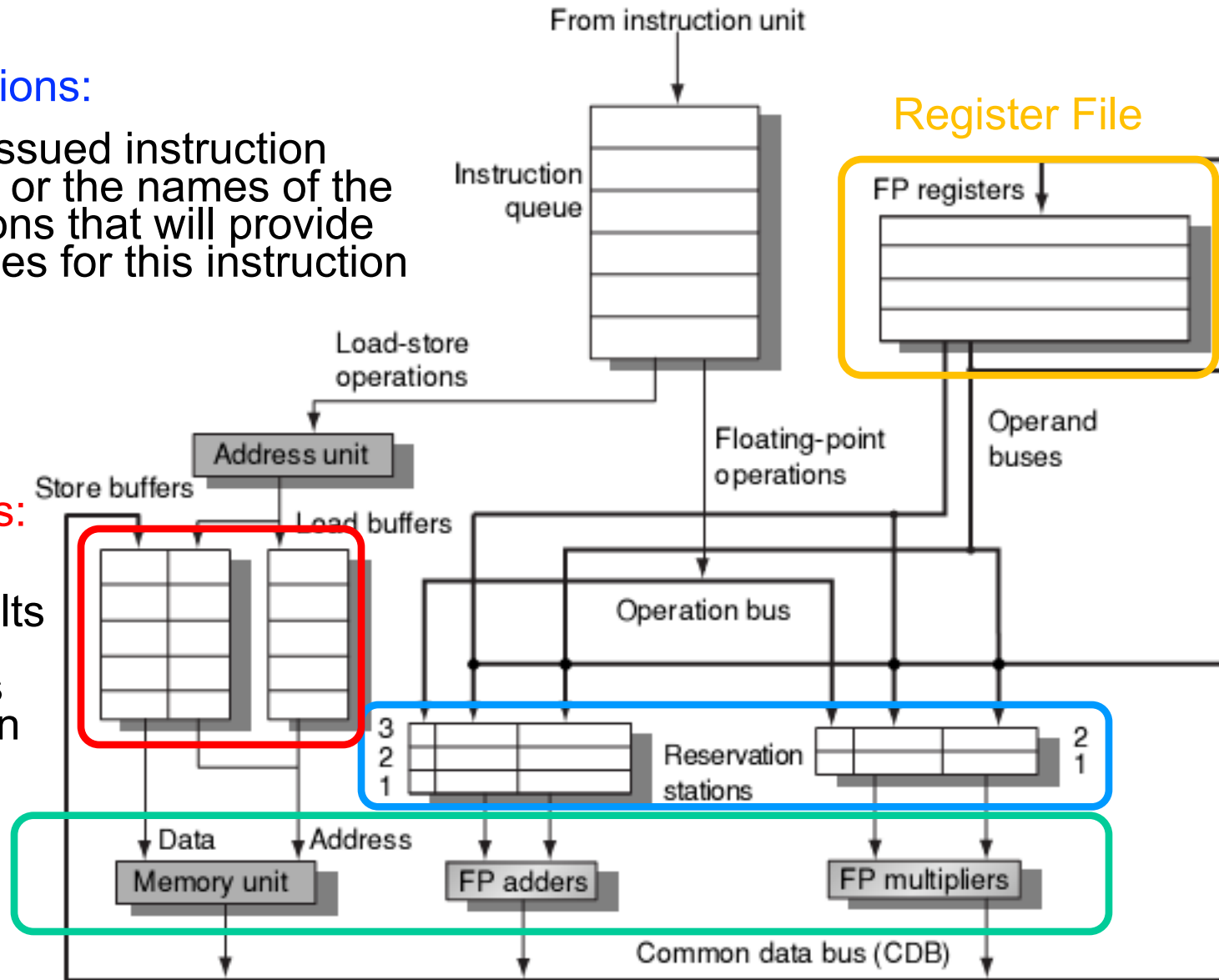
## Reservation Stations:

contain already issued instruction and its operands or the names of the reservation stations that will provide the operand values for this instruction

## Load/store buffers:

hold the effective address, the results of the completed loads, track loads that are waiting on the memory

## Functional Units:



Common Data Bus

# Τα στάδια του αλγορίθμου Tomasulo

**Εκδοση (Issue):** εισαγει την επόμενη εντολή από την ουρά εντολών

Αν υπάρχει ελεύθερο RS (**no structural hazard**), στείλε (**issue**) σε αυτό την εντολή, μαζί με τους operands (**rename registers**)

**Εκτέλεση (Execute):** εκτέλεση στην αριθμητική μονάδα (EX)

Όταν και οι δύο operands είναι διαθέσιμοι, τότε εκτέλεσε την πράξη. Αν δεν είναι διαθέσιμοι, παρακολούθησε το CDB για το αποτέλεσμα

**Εγγραφή Αποτελέσματος (Write result):** τέλος εκτέλεσης (WB)

Γράψε το αποτέλεσμα στο CDB για όλες τις μονάδες που το περιμένουν. Σημείωσε τον RS ως διαθέσιμο



# Περιγραφή Δομών (1)

- **Reservation Station fields**

- **Op**: λειτουργία προς εκτέλεση (π.χ. +, -, \*, ...)
- **Vj, Vk**: τιμές των source operands
- **Qj, Qk**: ποιά RS θα στείλουν την τιμή των source operands
  - » Σε οποιαδήποτε στιγμή, είτε το Q είτε το V είναι έγκυρο για κάποιον operand
- **Busy**: αν το RS είναι απασχολημένο ή όχι

Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
Add1	No					
Add2	No					
Add3	No					
Mult1	No					
Mult2	No					

# Περιγραφή Δομών (2)

- **Register Result status**

- **Qi** : Ποιο RS περιέχει την εντολή η οποία θα αποθηκεύσει το αποτέλεσμα στον register.



- **Load,Store Buffer fields**

- **A**: η effective address της θέσης μνήμης προς ανάγνωση/εγγραφή
- **Busy**: δείχνει αν ο buffer είναι απασχολημένος ή όχι

	<b>Busy</b>	<b>Address</b>
<b>Load1</b>	No	
<b>Load2</b>	No	
<b>Load3</b>	No	

# Περιγραφή Δομών (3)

- **Common Data Bus**

- Συνηθισμένα data bus: data + destination (“go to” bus)
- **CDB**: data + source (“come from” bus)
- 64 bits of data + 4 bits #RS
- Αν το **source** είναι ίδιο με το **Q** πεδίο ενός RS, γράψε το αποτέλεσμα στο αντίστοιχο **V** πεδίο του RS
- Broadcast: ένας «μιλάει», όλοι «ακούνε»

# Εκτέλεση εντολής κατά τον Αλγόριθμο Tomasulo

## 1. Issue — get instruction from FP Op Queue

*r = find free RS for this type of instructions (if no free RS => stall)*

$RS[r].op = op; RS[r].busy = 1;$

$RS[r].V_j = RF[rs].V; RS[r].Q_j = RF[rs].Q;$

$RS[r].V_k = RF[rt].V \quad RS[r].Q_k = RF[rt].Q$

$RF[rd].Q = r$

## 2. Wait — collect “interesting” results from CDB until ready

$\forall r, \text{ if } (r.Q_j = CDB.Q \text{ and } CDB.valid) \{ r.V_j = CDB.V; r.Q_j = 0 \}$

$\forall r, \text{ if } (r.Q_k = CDB.Q \text{ and } CDB.valid) \{ r.V_k = CDB.V; r.Q_k = 0 \}$

$\forall RFe, \text{ if } (RFe.Q = CDB.Q \text{ and } CDB.valid) \{ RFe.V_k = CDB.V; r.Q_k = 0 \}$

## 2. Execute — when ready, operate on operands (EX)

$\forall r, \text{ if } (r.Q_j = 0 \text{ and } r.Q_k = 0) \text{ and FU is free, start operation}$

## 3. Write result — finish execution (WB)

If CDB is free:  $\{ CDB.Q = r; CDB.V = value; CDB.valid = 1 \}$

Instruction state	Wait until	Action or bookkeeping
Issue FP operation	Station r empty	<pre> if (RegisterStat[rs].Qi≠0)     {RS[r].Qj ← RegisterStat[rs].Qi} else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0}; if (RegisterStat[rt].Qi≠0)     {RS[r].Qk ← RegisterStat[rt].Qi} else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0}; RS[r].Busy ← yes; RegisterStat[rd].Q ← r; </pre>
Load or store	Buffer r empty	<pre> if (RegisterStat[rs].Qi≠0)     {RS[r].Qj ← RegisterStat[rs].Qi} else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0}; RS[r].A ← imm; RS[r].Busy ← yes; </pre>
Load only		<pre> RegisterStat[rt].Qi ← r; </pre>
Store only		<pre> if (RegisterStat[rt].Qi≠0)     {RS[r].Qk ← RegisterStat[rs].Qi} else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0}; </pre>

Instruction state	Wait until	Action or bookkeeping
Execute FP operation	(RS[r].Qj = 0) and (RS[r].Qk = 0)	Compute result: operands are in Vj and Vk
Load-store step 1	RS[r].Qj = 0 & r is head of load-store queue	RS[r].A ← RS[r].Vj + RS[r].A;
Load step 2	Load step 1 complete	Read from Mem[RS[r].A]
Write Result FP operation or load	Execution complete at r & CDB available	$\forall x(\text{if } (\text{RegisterStat}[x].Qi=r) \{ \text{Regs}[x] \leftarrow \text{result};$ $\text{RegisterStat}[x].Qi \leftarrow 0 \});$ $\forall x(\text{if } (\text{RS}[x].Qj=r) \{ \text{RS}[x].Vj \leftarrow \text{result}; \text{RS}[x].Qj \leftarrow$ $0 \});$ $\forall x(\text{if } (\text{RS}[x].Qk=r) \{ \text{RS}[x].Vk \leftarrow \text{result}; \text{RS}[x].Qk \leftarrow$ $0 \});$ RS[r].Busy ← no;
Store	Execution complete at r & RS[r].Qk = 0	Mem[RS[r].A] ← RS[r].Vk; RS[r].Busy ← no;

# Tomasulo Example

Instruction stream



Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2			
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

3 Load/Buffers

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

FU count  
down

3 FP Adder R.S.  
2 FP Mult R.S.

Register result status:

Clock



0

Clock cycle  
counter

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

(load: 2 cycles, add: 2 cycles, mult: 10 cycles, divide 40 cycles)

# Tomasulo Example Cycle 1

## Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1			Yes	34+R2
LD	F2	45+	R3				No	
MULTD	F0	F2	F4				No	
SUBD	F8	F6	F2					
DIVD	F10	F0	F6					
ADDD	F6	F8	F2					

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
1				Load1					



# Tomasulo Example Cycle 2

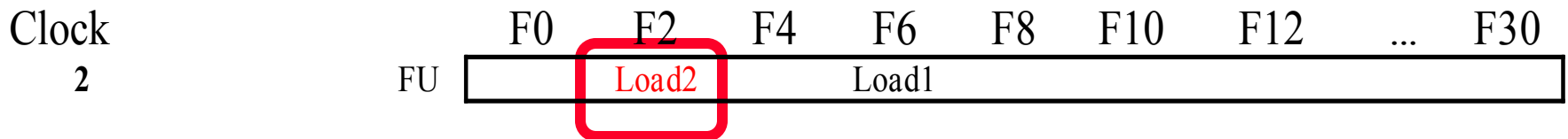
## Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1			Load1	Yes 34+R2
LD	F2	45+	R3	2			Load2	Yes 45+R3
MULTD	F0	F2	F4				Load3	No
SUBD	F8	F6	F2					
DIVD	F10	F0	F6					
ADDD	F6	F8	F2					

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

## Register result status:



# Tomasulo Example Cycle 3

## Instruction status:

Instruction	j	k	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1	3	Load1	Yes 34+R2
LD	F2	45+	R3	2		Load2	Yes 45+R3
MULTD	F0	F2	F4	3		Load3	No
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		R(F4)	Load2	
	Mult2	No					

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
3	Mult1	Load2			Load1				

- μόλις η εντολή γίνεται issue σε κάποιον RS, τα ονόματα των source registers αντικαθιστώνται (“renamed”) μέσω των πεδίων V ή Q του RS
- η εντολή στον Load1 ολοκληρώνεται - ποιος περιμένει για το αποτέλεσμα?

# Tomasulo Example Cycle 4

## Instruction status:

Instruction	j	k	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1	3	No	
LD	F2	45+	R3	2	4	Yes	45+R3
MULTD	F0	F2	F4	3		No	
SUBD	F8	F6	F2	4			
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	Yes	SUBD	M(A1)			Load2
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		R(F4)		Load2
	Mult2	No					

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
4	Mult1	Load2		M(A1)	Add1				

- η εντολή στον Load2 ολοκληρώνεται - ποιος περιμένει για το αποτέλεσμα?

# Tomasulo Example Cycle 5

## Instruction status:

Instruction	j	k	Issue	Exec Comp	Write Result	Busy	Address	
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4				
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2					

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
2	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	No					
	Add3	No					
10	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
5	Mult1	M(A2)		M(A1)	Add1	Mult2			

- αρχίζει η αντίστροφη μέτρηση για τους Add1, Mult1 (load: 1 cycle, add: 2 cycles, mult: 10 cycles, divide 40 cycles)

# Tomasulo Example Cycle 6

## Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result		Busy	Address
LD	F6	34+	R2	1	3	4	Load1	No	
LD	F2	45+	R3	2	4	5	Load2	No	
MULTD	F0	F2	F4	3			Load3	No	
SUBD	F8	F6	F2	4					
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6					

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
1	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
9	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
6	Mult1	M(A2)		Add2	Add1	Mult2			

- η ADDD γίνεται issue εδώ παρά την name dependency στον F6

# Tomasulo Example Cycle 7

## Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result		Busy	Address
LD	F6	34+	R2	1	3	4	Load1	No	
LD	F2	45+	R3	2	4	5	Load2	No	
MULTD	F0	F2	F4	3			Load3	No	
SUBD	F8	F6	F2	4	7				
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6					

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
0	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
7	Mult1	M(A2)		Add2	Add1	Mult2			

- η εντολή στον Add1 (SUBD) ολοκληρώνεται - ποιος περιμένει για το αποτέλεσμα?

# Tomasulo Example Cycle 8

## Instruction status:

Instruction	j	k	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4	5	Load2
MULTD	F0	F2	F4	3			Load3
SUBD	F8	F6	F2	4	7	8	
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6			

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
2	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
7	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
8	Mult1	M(A2)		Add2	(M-M)	Mult2			

# Tomasulo Example Cycle 9

## Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6				

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
1	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
6	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
9	Mult1	M(A2)		Add2	(M-M)	Mult2			



# Tomasulo Example Cycle 10

## Instruction status:

Instruction	j	k	Issue	Exec Comp	Write Result	Busy	Address	
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10			

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
0	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
5	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
10	Mult1	M(A2)		Add2	(M-M)	Mult2			

- η εντολή στον Add2 (ADDD) ολοκληρώνεται - ποιος περιμένει για το αποτέλεσμα?

# Tomasulo Example Cycle 11

## Instruction status:

Instruction	j	k	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4	5	Load2
MULTD	F0	F2	F4	3			Load3
SUBD	F8	F6	F2	4	7	8	
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6	10	11	

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
11	FU	Mult1	M(A2)	(M-M+M)	(M-M)	Mult2			

- η ADDD γράφει το αποτέλεσμα της

# Tomasulo Example Cycle 12

## Instruction status:

Instruction	j	k	Issue	Exec Comp	Write Result	Busy	Address	
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
3	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
12	Mult1	M(A2)		(M-M+M)	(M-M)	Mult2			

# Tomasulo Example Cycle 13

## Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result		Busy	Address
LD	F6	34+	R2	1	3	4	Load1	No	
LD	F2	45+	R3	2	4	5	Load2	No	
MULTD	F0	F2	F4	3			Load3	No	
SUBD	F8	F6	F2	4	7	8			
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6	10	11			

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
2	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
13	Mult1	M(A2)		(M-M+M)	(M-M)	Mult2			

# Tomasulo Example Cycle 14

## Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
14	Mult1	M(A2)		(M-M+M)	(M-M)	Mult2			

# Tomasulo Example Cycle 15

## Instruction status:

Instruction	j	k	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4	5	Load2
MULTD	F0	F2	F4	3	15		Load3
SUBD	F8	F6	F2	4	7	8	
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6	10	11	

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
15	Mult1	M(A2)		(M-M+M)	(M-M)	Mult2			

- η εντολή στον Mult1 (MULTD) ολοκληρώνεται - ποιος περιμένει για το αποτέλεσμα?

# Tomasulo Example Cycle 16

## Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15	16	Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
40	Mult2	Yes	DIVD	M*F4	M(A1)		

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
16	M*F4	M(A2)		(M-M+M)	(M-M)	Mult2			

- ...εκκρεμεί πλέον μόνο η DIVD (div: 40 cycles)

Μετά από κάμπουσους κύκλους...



# Tomasulo Example Cycle 55

## Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result		Busy	Address
LD	F6	34+	R2	1	3	4	Load1	No	
LD	F2	45+	R3	2	4	5	Load2	No	
MULTD	F0	F2	F4	3	15	16	Load3	No	
SUBD	F8	F6	F2	4	7	8			
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6	10	11			

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
1	Mult2	Yes	DIVD	M*F4	M(A1)		

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
55	M*F4	M(A2)		(M-M+M)	(M-M)	Mult2			

# Tomasulo Example Cycle 56

## Instruction status:

Instruction	j	k	Issue	Exec Comp	Write Result	Busy	Address	
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15	16	Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5	56			
ADDD	F6	F8	F2	6	10	11		

## Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	Yes	DIVD	M*F4	M(A1)		

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
56	M*F4	M(A2)		(M-M+M)	(M-M)	Mult2			

- η εντολή στον Mult2 (DIVD) ολοκληρώνεται - ποιος περιμένει για το αποτέλεσμα?

# Tomasulo Example Cycle 57

## Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15	16	Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5	56	57		
ADD	F6	F8	F2	6	10	11		

## Reservation Stations:

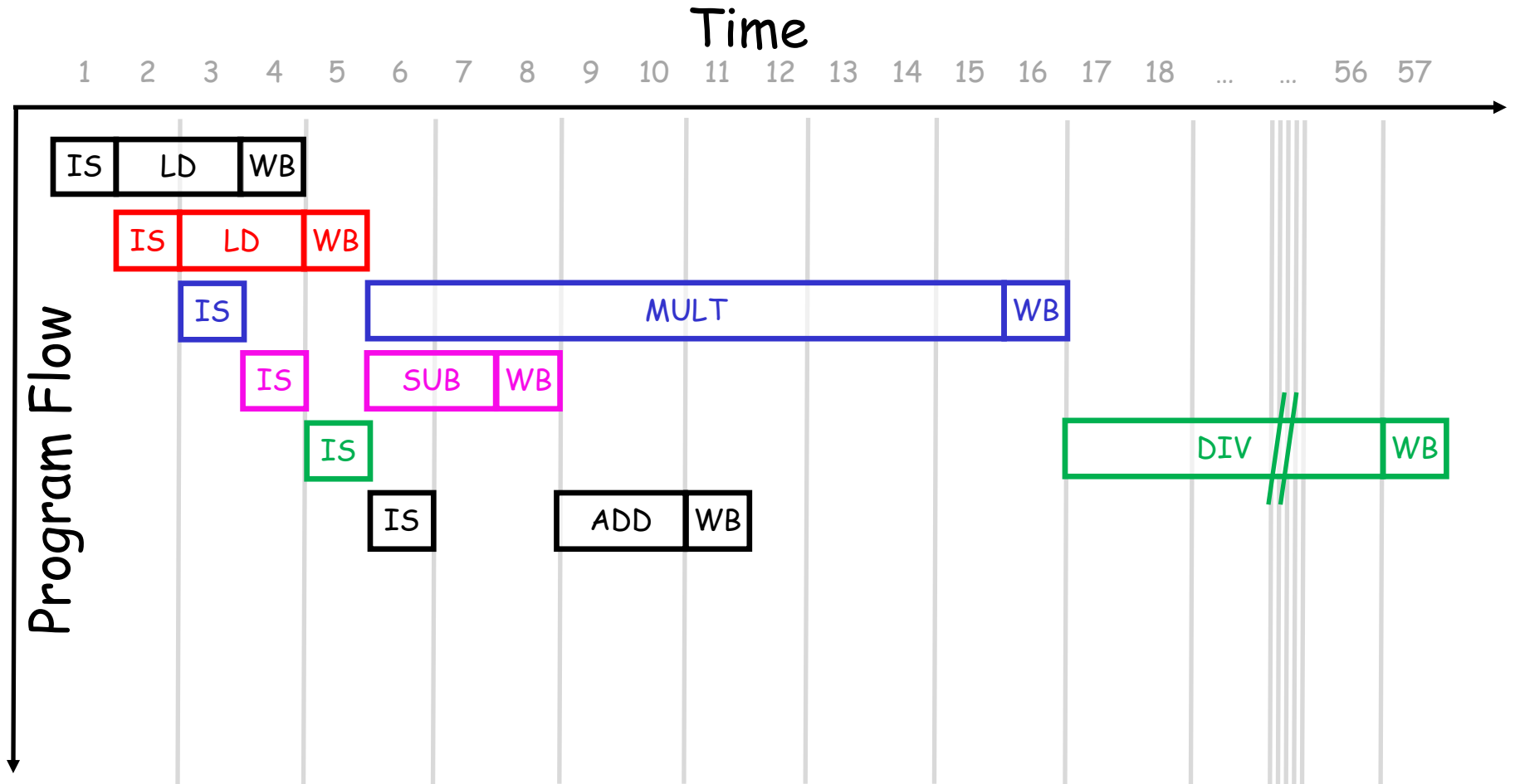
Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	Yes	DIVD	M*F4	M(A1)		

## Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
56	M*F4	M(A2)		(M-M+M)	(M-M)	Result			

- Συνοψίζοντας: **In-order issue, out-of-order execution και out-of-order completion.**

# Tomasulo Dynamic Execution



# Tomasulo Loop Example

Loop: LD	F0	0	R1
MULTD	F4	F0	F2
SD	F4	0	R1
SUBI	R1	R1	#8
BNEZ	R1	Loop	

- mult: 4 cycles
- 1st load: 8 cycles (L1 cache miss)
- 2nd load: 4 cycles (hit)
- to branch προβλέπεται σαν TAKEN

# Loop Example

## Instruction status:

ITER	Instruction	j	k	Issue	Comp	Result
1	LD	F0	0	R1		
1	MULTD	F4	F0	F2		
1	SD	F4	0	R1		
2	LD	F0	0	R1		
2	MULTD	F4	F0	F2		
2	SD	F4	0	R1		

Iteration Count

Exec Write

Issue	Comp	Result

	Busy	Addr	Fu
Load1	No		
Load2	No		
Load3	No		
Store1	No		
Store2	No		
Store3	No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Code:

LD	F0	0	R1
MULTD	F4	F0	F2
SD	F4	0	R1
SUBI	R1	R1	#8
BNEZ	R1	Loop	

προσθέσαμε Store Buffers

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
0	80									

Instruction Loop

τιμή καταχωρητή που χρησιμοποιείται για διευθύνσεις και επαναλήψεις

# Loop Example Cycle 1

## Instruction status:

ITER	Instruction	Fu	j	k	Issue	Exec	Write
1	LD	F0	0	R1	1	Comp	Result

	Busy	Addr	Fu
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	No		
Store2	No		
Store3	No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD
	Add2	No						MULTD
	Add3	No						SD
	Mult1	No						SUBI
	Mult2	No						BNEZ

S1	S2	RS

Code:	Fu	Op	Rj	Rk
LD	F0	0	R1	
MULTD	F4	F0	F2	
SD	F4	0	R1	
SUBI	R1	R1	#8	
BNEZ	R1	Loop		

## Register result status

Clock	R1	Fu	F0	F2	F4	F6	F8	F10	F12	...	F30
1	80	Load1									

# Loop Example Cycle 2

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu
1	LD	F0	0	R1	1		Yes	80	
1	MULTD	F4	F0	F2	2		No		
							No		
							No		
							No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD
	Add2	No						MULTD
	Add3	No						SD
	Mult1	Yes	Multd		R(F2)	Load1		SUBI
	Mult2	No						BNEZ

Code: LD F0 0 R1  
MULTD F4 F0 F2  
SD F4 0 R1  
SUBI R1 R1 #8  
BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
2	80	Load1		Mult1						



# Loop Example Cycle 3

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu	
					Comp	Result				
1	LD	F0	0	R1	1		Load1	Yes	80	
1	MULTD	F4	F0	F2	2		Load2	No		
1	SD	F4	0	R1	3		Load3	No		
							Store1	Yes	80	Mult1
							Store2	No		
							Store3	No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	S1	S2	RS	Code:
						Op	Qj	Qk	
	Add1	No							LD
	Add2	No							MULTD
	Add3	No							SD
	Mult1	Yes	Multd		R(F2)		Load1		SUBI
	Mult2	No							BNEZ

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
3	80	Fu	Load1	Mult1						

# Loop Example Cycle 4

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu
1	LD	F0	0	R1	1		Yes	80	
1	MULTD	F4	F0	F2	2		No		
1	SD	F4	0	R1	3		No		
							Yes	80	Mult1
							No		
							No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	S1	S2	RS	Code:
	Add1	No			S1	S2	RS <td>LD F0 0 R1</td>	LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1
	Mult1	Yes	Multd		R(F2)	Load1		SUBI R1 R1 #8
	Mult2	No						BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
4	80	Fu	Load1	Mult1						

- (η εντολή SUBI -δε βρίσκεται στην FP queue- γίνεται dispatch)

# Loop Example Cycle 5

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu
1	LD	F0	0	R1	1		Yes	80	
1	MULTD	F4	F0	F2	2		No		
1	SD	F4	0	R1	3		No		
							Yes	80	Mult1
							No		
							No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1
	Mult1	Yes	Multd		R(F2)	Load1		SUBI R1 R1 #8
	Mult2	No						BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
5	72	Load1		Mult1						

- (το ίδιο και η BNEZ)

# Loop Example Cycle 6

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu
1	LD	F0	0	R1	1		Load1	Yes	80
1	MULTD	F4	F0	F2	2		Load2	Yes	72
1	SD	F4	0	R1	3		Load3	No	
2	LD	F0	0	R1	6		Store1	Yes	80
							Store2	No	
							Store3	No	
									Mult1

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	S1	S2	RS	Code:
	Add1	No							LD
	Add2	No							MULTD
	Add3	No							SD
	Mult1	Yes	Multd		R(F2)	Load1			SUBI
	Mult2	No							BNEZ

Code: LD F0 0 R1  
MULTD F4 F0 F2  
SD F4 0 R1  
SUBI R1 R1 #8  
BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
6	72	Load2		Mult1						

- Ο F0 ποτέ δεν “βλέπει” κάποιο load από τη θέση 80

# Loop Example Cycle 7

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu	
1	LD	F0	0	R1	1		Load1	Yes	80	
1	MULTD	F4	F0	F2	2		Load2	Yes	72	
1	SD	F4	0	R1	3		Load3	No		
2	LD	F0	0	R1	6		Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7		Store2	No		
							Store3	No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2 ←
	Add3	No						SD F4 0 R1
	Mult1	Yes	Multd		R(F2)	Load1		SUBI R1 R1 #8
	Mult2	Yes	Multd		R(F2)	Load2		BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
7	72	Fu	Load2	Mult2						

- το register file είναι αποσυνδεδεμένο από τους υπολογισμούς
- η 1<sup>η</sup> και η 2<sup>η</sup> επανάληψη επικαλύπτονται πλήρως

# Loop Example Cycle 8

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu	
1	LD	F0	0	R1	1		Load1	Yes	80	
1	MULTD	F4	F0	F2	2		Load2	Yes	72	
1	SD	F4	0	R1	3		Load3	No		
2	LD	F0	0	R1	6		Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7		Store2	Yes	72	Mult2
2	SD	F4	0	R1	8		Store3	No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1
	Mult1	Yes	Multd		R(F2)	Load1		SUBI R1 R1 #8
	Mult2	Yes	Multd		R(F2)	Load2		BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
8	72	Fu	Load2	Mult2						

# Loop Example Cycle 9

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu	
1	LD	F0	0	R1	1	9	Load1	Yes	80	
1	MULTD	F4	F0	F2	2		Load2	Yes	72	
1	SD	F4	0	R1	3		Load3	No		
2	LD	F0	0	R1	6		Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7		Store2	Yes	72	Mult2
2	SD	F4	0	R1	8		Store3	No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	S1	S2	RS	Code:			
					Vk	Qj	Qk				
	Add1	No						LD	F0	0	R1
	Add2	No						MULTD	F4	F0	F2
	Add3	No						SD	F4	0	R1
	Mult1	Yes	Multd		R(F2)	Load1		SUBI	R1	R1	#8
	Mult2	Yes	Multd		R(F2)	Load2		BNEZ	R1	Loop	

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
9	72	Fu	Load2	Mult2						

- η εντολή στον Load1 ολοκληρώνεται - ποιος περιμένει για το αποτέλεσμα?
- (η SUBI γίνεται dispatch)

# Loop Example Cycle 10

## Instruction status:

ITER	Instruction	j	k	Exec Write			Busy	Addr	Fu	
				Issue	Comp	Result				
1	LD	F0	0	R1	1	9	10	Load1	No	
1	MULTD	F4	F0	F2	2			Load2	Yes	72
1	SD	F4	0	R1	3			Load3	No	
2	LD	F0	0	R1	6	10		Store1	Yes	80
2	MULTD	F4	F0	F2	7			Store2	Yes	72
2	SD	F4	0	R1	8			Store3	No	

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
								S1	S2	RS	
	Add1	No						LD	F0	0	R1
	Add2	No						MULTD	F4	F0	F2
	Add3	No						SD	F4	0	R1
4	Mult1	Yes	Multd	M[80]	R(F2)			SUBI	R1	R1	#8
	Mult2	Yes	Multd		R(F2)	Load2		BNEZ	R1	Loop	

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
10	64	Load2		Mult2						

- η εντολή στον Load2 ολοκληρώνεται - ποιος περιμένει για το αποτέλεσμα?
- (η BNEZ γίνεται dispatch)



# Loop Example Cycle 11

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10		
1	MULTD	F4	F0	F2	2				
1	SD	F4	0	R1	3				
2	LD	F0	0	R1	6	10	11		
2	MULTD	F4	F0	F2	7				
2	SD	F4	0	R1	8				
	Load1						No		
	Load2						No		
	Load3						Yes	64	
	Store1						Yes	80	Mult1
	Store2						Yes	72	Mult2
	Store3						No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1
3	Mult1	Yes	Multd	M[80]	R(F2)			SUBI R1 R1 #8
4	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
11	64	64								

- επόμενο load στην ακολουθία

# Loop Example Cycle 12

## Instruction status:

ITER	Instruction	j	k	Exec Write			Busy	Addr	Fu
				Issue	Comp	Result			
1	LD	F0	0	R1	1	9	10	No	
1	MULTD	F4	F0	F2	2			No	
1	SD	F4	0	R1	3			Yes	64
2	LD	F0	0	R1	6	10	11	Yes	80
2	MULTD	F4	F0	F2	7			Yes	72
2	SD	F4	0	R1	8			No	

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2 ←
	Add3	No						SD F4 0 R1
2	Mult1	Yes	Multd	M[80]	R(F2)			SUBI R1 R1 #8
3	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
12	64	Fu	Load3	Mult2						

- γιατί να μην κάνουμε issue και τρίτο mult?

# Loop Example Cycle 13

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Result	Busy	Addr	Fu	
1	LD	F0	0	R1	1	9	10	Load1	No		
1	MULTD	F4	F0	F2	2			Load2	No		
1	SD	F4	0	R1	3			Load3	Yes	64	
2	LD	F0	0	R1	6	10	11	Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7			Store2	Yes	72	Mult2
2	SD	F4	0	R1	8			Store3	No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1
1	Mult1	Yes	Multd	M[80]	R(F2)			SUBI R1 R1 #8
2	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
13	64	Fu	Load3	Mult2						

- γιατί να μην κάνουμε issue και τρίτο store?

# Loop Example Cycle 14

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	No		
1	MULTD	F4	F0	F2	2	14	No		
1	SD	F4	0	R1	3		Yes	64	
2	LD	F0	0	R1	6	10	Yes	80	Mult1
2	MULTD	F4	F0	F2	7		Yes	72	Mult2
2	SD	F4	0	R1	8		No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	S1	S2	RS	Code:
	Add1	No							LD
	Add2	No							MULTD
	Add3	No							SD
0	Mult1	Yes	Multd	M[80]	R(F2)				SUBI
1	Mult2	Yes	Multd	M[72]	R(F2)				BNEZ

Code: LD F0 0 R1  
MULTD F4 F0 F2  
SD F4 0 R1  
SUBI R1 R1 #8  
BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
14	64	Fu	Load3	Mult2						

- η εντολή στον Mult1 ολοκληρώνεται - ποιος περιμένει για το αποτέλεσμα?

# Loop Example Cycle 15

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	No		
1	MULTD	F4	F0	F2	2	14	No		
1	SD	F4	0	R1	3		Yes	64	
2	LD	F0	0	R1	6	10	Yes	80	[80]*R2
2	MULTD	F4	F0	F2	7	15	Yes	72	Mult2
2	SD	F4	0	R1	8		No		

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2 ←
	Add3	No						SD F4 0 R1
	Mult1	No						SUBI R1 R1 #8
0	Mult2	Yes	Multd	M[72]	R(F2)			BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
15	64	Fu	Load3	Mult2						

- η εντολή στον Mult2 ολοκληρώνεται - ποιος περιμένει για το αποτέλεσμα?

# Loop Example Cycle 16

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10		
1	MULTD	F4	F0	F2	2	14	15		
1	SD	F4	0	R1	3				
2	LD	F0	0	R1	6	10	11		
2	MULTD	F4	F0	F2	7	15	16		
2	SD	F4	0	R1	8				
Load1		No							
Load2		No							
Load3		Yes	64						
Store1		Yes	80						[80]*R2
Store2		Yes	72						[72]*R2
Store3		No							

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2 ←
	Add3	No						SD F4 0 R1
4	Mult1	Yes	Multd		R(F2)	Load3		SUBI R1 R1 #8
	Mult2	No						BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
16	64	Fu	Load3	Mult1						

- Γίνεται issue to 3<sup>ο</sup> MULTD

# Loop Example Cycle 17

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3			Load3	Yes 64
2	LD	F0	0	R1	6	10	11	Store1	Yes 80 [80]*R2
2	MULTD	F4	F0	F2	7	15	16	Store2	Yes 72 [72]*R2
2	SD	F4	0	R1	8			Store3	Yes 64 <b>Mult1</b>

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1 ←
	Mult1	Yes	Multd		R(F2)	Load3		SUBI R1 R1 #8
	Mult2	No						BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
17	64	Fu	Load3	Mult1						

- ...μπορεί να γίνει issue και το 3<sup>ο</sup> SD

# Loop Example Cycle 18

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10	Load1	No
1	MULTD	F4	F0	F2	2	14	15	Load2	No
1	SD	F4	0	R1	3	18		Load3	Yes
2	LD	F0	0	R1	6	10	11	Store1	Yes
2	MULTD	F4	F0	F2	7	15	16	Store2	Yes
2	SD	F4	0	R1	8			Store3	Yes

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD
	Add2	No						MULTD
	Add3	No						SD
	Mult1	Yes	Multd		R(F2)	Load3		SUBI
	Mult2	No						BNEZ

Code: LD F0 0 R1  
MULTD F4 F0 F2  
SD F4 0 R1  
SUBI R1 R1 #8  
BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
18	64	Fu	Load3	Mult1						

- ...ολοκληρώνεται η εκτέλεση του 1<sup>ου</sup> SD



# Loop Example Cycle 19

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu
1	LD	F0	0	R1	1	9	10		
1	MULTD	F4	F0	F2	2	14	15		
1	SD	F4	0	R1	3	18	19		
2	LD	F0	0	R1	6	10	11		
2	MULTD	F4	F0	F2	7	15	16		
2	SD	F4	0	R1	8	19			
Load1		No							
Load2		No							
Load3		Yes	64						
Store1		No							
Store2		Yes	72						[72]*R2
Store3		Yes	64						Mult1

## Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1
	Mult1	Yes	Multd		R(F2)	Load3		SUBI R1 R1 #8
	Mult2	No						BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
19	56	Fu	Load3	Mult1						

- ...ολοκληρώνεται η εκτέλεση του 2<sup>ου</sup> SD

# Loop Example Cycle 20

## Instruction status:

ITER	Instruction	j	k	Issue	Exec	Write	Busy	Addr	Fu		
1	LD	F0	0	R1	1	9	10	Load1	Yes	56	
1	MULTD	F4	F0	F2	2	14	15	Load2	No		
1	SD	F4	0	R1	3	18	19	Load3	Yes	64	
2	LD	F0	0	R1	6	10	11	Store1	No		
2	MULTD	F4	F0	F2	7	15	16	Store2	No		
2	SD	F4	0	R1	8	19	20	Store3	Yes	64	Mult1

## Reservation Stations:

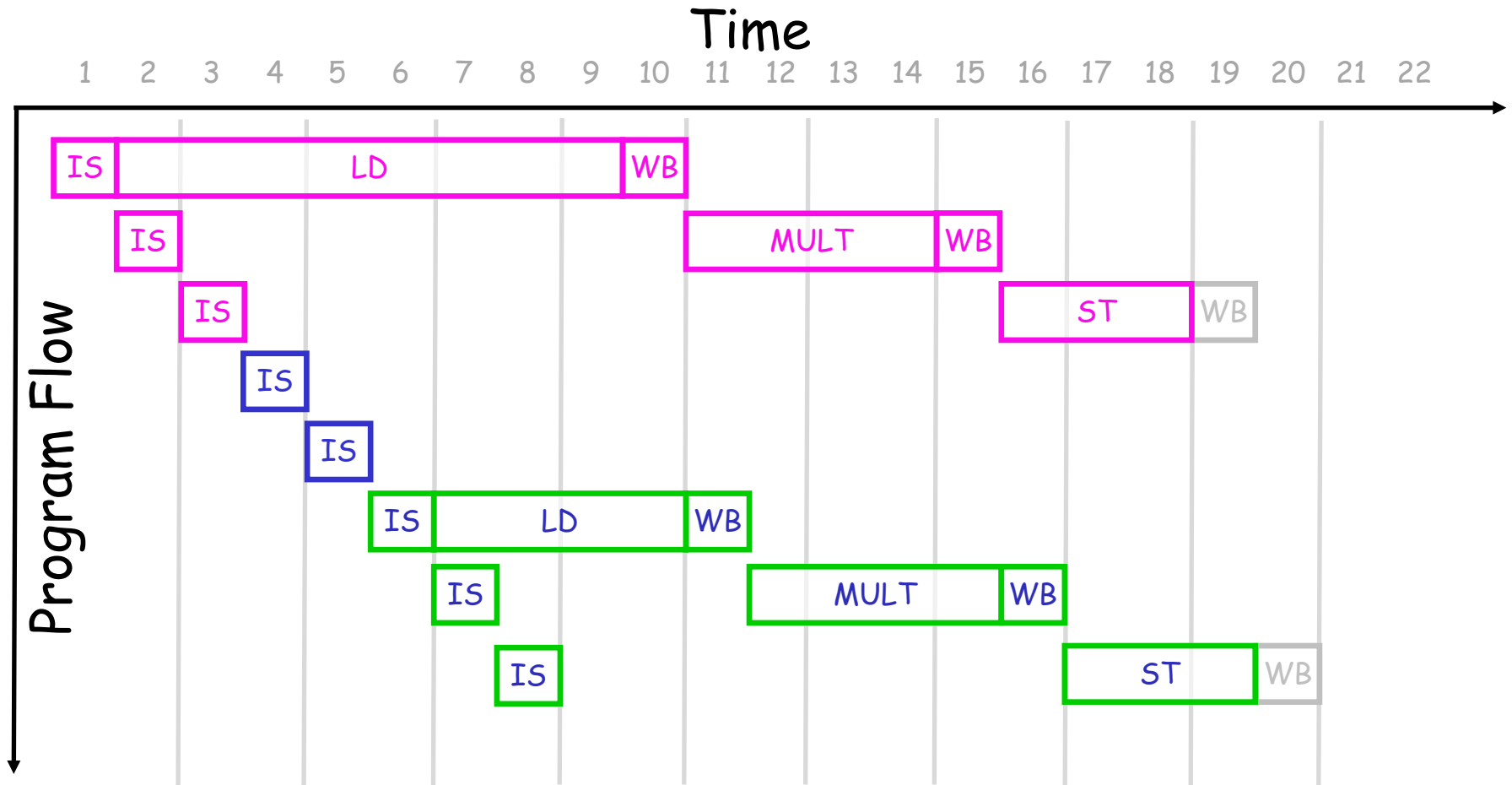
Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2 ←
	Add3	No						SD F4 0 R1
	Mult1	Yes	Multd		R(F2)	Load3		SUBI R1 R1 #8
	Mult2	No						BNEZ R1 Loop

## Register result status

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
20	56	Fu	Load1	Mult1						

- Συνοψίζοντας για άλλη μια φορά: **In-order issue**, **out-of-order execution** και **out-of-order completion**

# Tomasulo Dynamic Execution



# Γιατί τελικά καταφέρνει ο αλγόριθμος να επικαλύψει τις επαναλήψεις?

- Register renaming
  - διαδοχικές επαναλήψεις χρησιμοποιούν διαφορετικούς «φυσικούς προορισμούς» ως καταχωρητές προορισμού (dynamic loop unrolling)
- Reservation stations
  - επιτρέπουν την έκδοση εντολών να προχωρήσει μπροστά από την εκτέλεση (πέρα από εξαρτήσεις δεδομένων και ελέγχου)
  - κάνουν buffer τις παλιές τιμές των registers – αποφεύγονται πλήρως τα stalls εξαιτίας των WAR hazards

# Βασικά πλεονεκτήματα του αλγορίθμου

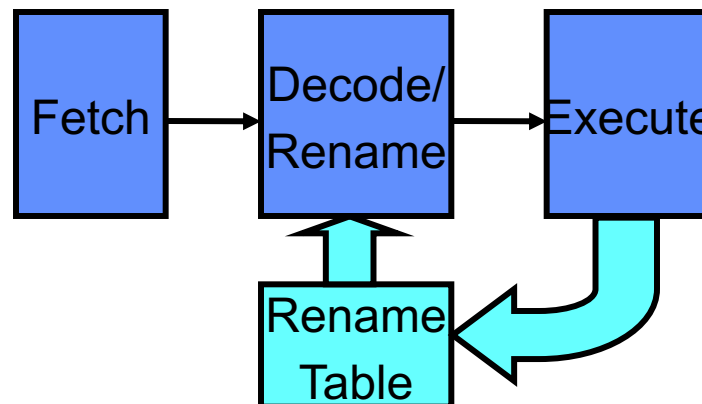
- Κατανεμημένη λογική για την ανίχνευση των hazards
  - Κατανεμημένος έλεγχος για κάθε reservation station
  - αν πολλές εντολές περιμένουν 1 αποτέλεσμα (και έχουν τον άλλον operand διαθέσιμο), τότε μπορούν να «απελευθερωθούν» **ταυτόχρονα** όταν αυτό γίνει broadcast στο CDB
    - » αν χρησιμοποιούταν ένα κεντρικό register file, τότε οι μονάδες εκτέλεσης θα έπρεπε να διαβάζουν από εκεί τα δεδομένα τους, κάθε φορά που θα τους παραχωρείτο το register bus (πολλές θύρες ανάγνωσης στην RF)
- Αποφυγή stalls εξαιτίας των WAW και WAR hazards
  - Με την χρήση μετονομασίας των καταχωρητών στις αντίστοιχες θέσεις reservation station

# Explicit Register Renaming

- **ΙΔΕΑ** : Γιατί να μην έχω “άπειρους” καταχωρητές ώστε να μπορώ να εκτελώ πιο εύκολα και αποδοτικά register renaming;
- Απαιτείται :
  - Ένα physical register file (PRF) με **περισσότερους** «φυσικούς» καταχωρητές από όσους ορίζει η ISA
  - Translation Table (γρήγορα προσπελάσιμος)
    - » Λογικό όνομα καταχωρητή, φυσική θέση που αντιστοιχεί!
  - Μηχανισμός εντοπισμού ελεύθερων φυσικών καταχωρητών

## Explicit Register Renaming(2)

- Το pipeline μπορεί να παραμείνει ίδιο με το κλασσικό 5-stage pipeline



- Κατά το decode κάθε ISA register που χρησιμοποιείται ως όρισμα της εντολής αντιστοιχίζεται σε έναν φυσικό καταχωρητή
  - » **target** : Επιλέγεται ένας από τους ελεύθερους καταχωρητές και η αντιστοίχιση αυτή αποθηκεύεται στο Register Map Table (RMT)
  - » **source** : Χρησιμοποιείται η τελευταία αντιστοίχιση που είναι αποθηκευμένη στο RMT
- Κάθε φυσικός καταχωρητής που δεν χρησιμοποιείται από καμιά εντολή σε εκτέλεση, θεωρείται ελεύθερος.

# Παράδειγμα

## Instruction Stream

**DIV** R5 , R4 , R2  
**ADD** R7 , R5 , R1  
**SUB** R5 , R3 , R2  
**LD** R7 , 1000 (R5)

## Register Map Table

R1	PR23
R2	PR2
R3	PR17
R4	PR45
R5	PR13
R6	PR20
R7	PR30
...	...

## Free Registers

**PR37,PR4,PR42,PR19,...**



# Παράδειγμα (1)

## Instruction Stream

**DIV** R5, R4, R2  
**ADD** R7, R5, R1  
**SUB** R5, R3, R2  
**LD** R7, 1000 (R5)

## Register Map Table

R1	PR23
R2	PR2
R3	PR17
R4	PR45
R5	PR13
R6	PR20
R7	PR30
...	...

## Free Registers

PR37, PR4, PR42, PR19, ...

**DIV** PR37, PR45, PR2

# Παράδειγμα (2)

## Instruction Stream

DIV R5 , R4 , R2  
ADD R7 , R5 , R1  
SUB R5 , R3 , R2  
LD R7 , 1000 (R5)

## Register Map Table

R1	PR23
R2	PR2
R3	PR17
R4	PR45
R5	PR37
R6	PR20
R7	PR30
...	...

## Free Registers

PR4, PR42, PR19, ...



DIV PR37 , PR45 , PR2  
ADD PR4 , PR37 , PR23

# Παράδειγμα (3)

## Instruction Stream

```

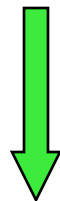
DIV   R5 , R4 , R2
ADD   R7 , R5 , R1
SUB  R5 , R3 , R2
LD    R7 , 1000 (R5)
    
```

## Register Map Table

R1	PR23
R2	<b>PR2</b>
R3	<b>PR17</b>
R4	PR45
R5	<b>PR37</b>
R6	PR20
R7	PR4
...	...

## Free Registers

**PR42, PR19, ...**



```

DIV   PR37 , PR45 , PR2
ADD   PR4  , PR37 , PR23
SUB  PR42 , PR17 , PR2
    
```

# Παράδειγμα (4)

## Instruction Stream

```

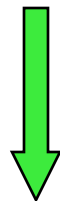
DIV   R5 , R4 , R2
ADD   R7 , R5 , R1
SUB   R5 , R3 , R2
LD   R7 , 1000 (R5)
    
```

## Register Map Table

R1	PR23
R2	PR2
R3	PR17
R4	PR45
R5	<b>PR42</b>
R6	PR20
R7	<b>PR4</b>
...	...

## Free Registers

**PR19,...**



```

DIV   PR37 , PR45 , PR2
ADD   PR4  , PR37 , PR23
SUB   PR42 , PR17 , PR2
LD   PR19 , 1000 (PR42)
    
```

# Πλεονεκτήματα

- Δε χρειάζεται reservation stations
- Αποσύνδεση του **renaming** από το **scheduling** των εντολών
  - Το pipeline μπορεί να παραμείνει απλό όπως το κλασικό 5-stage pipeline
- Τα δεδομένα μεταφέρονται από ένα μοναδικό register file
- Αποφυγή όλων των WAR, WAW hazards
- Επιτρέπει (όπως και ο Tomasulo) out-of-order completion
- Πολλές σύγχρονες αρχιτεκτονικές χρησιμοποιούν συνδυασμό **explicit register renaming** + **Tomasulo**