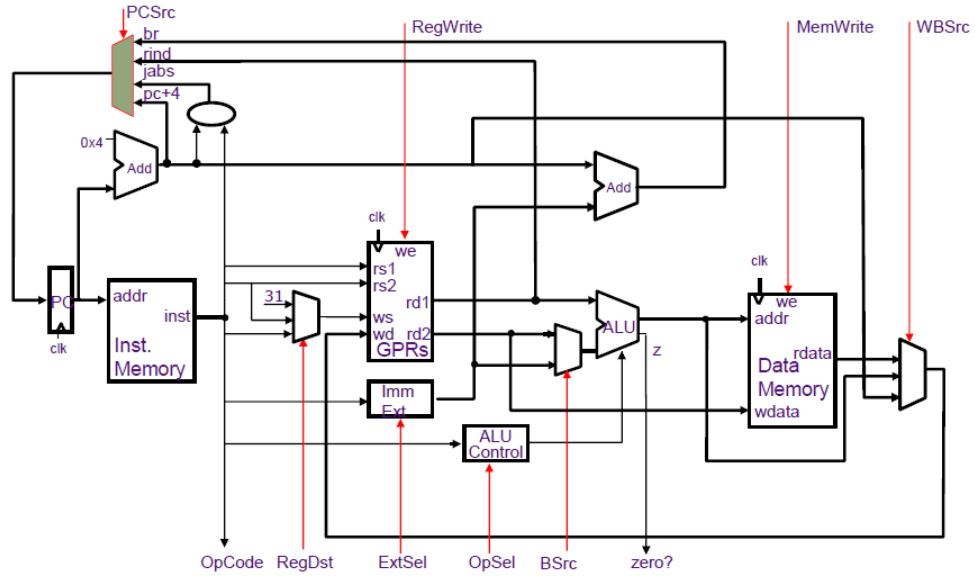
Pipelining Εντολών

Μάρτιος 2023

Datapath Εντολών σε Ένα Κύκλο Ρολογιού

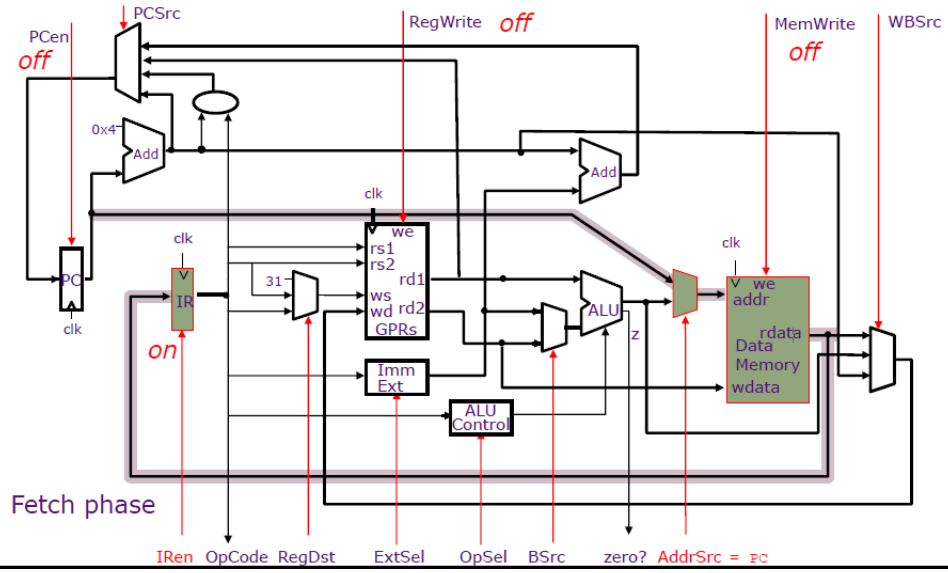


Κίνδυνος Σχεδιαστικός

- Τι πρόβλημα προκύπτει αν υπάρχει μία μοναδική μνήμη για να αποθηκεύουμε εντολές και δεδομένα?
  - Το διάβασμα μιας εντολής και η αναφορά στην μνήμη για μια μεταβλητή (διάβασμα ή αποθήκευση) δεν μπορούν να εκτελεστούν στον ίδιο κύκλο ρολογιού.

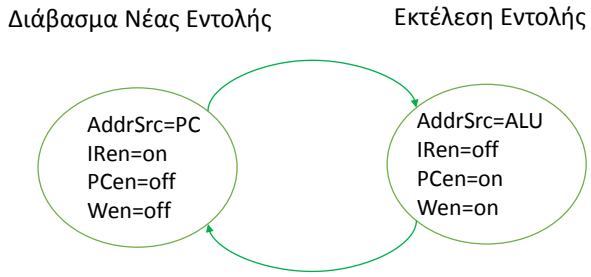
3

Έλεγχος και Datapath



4

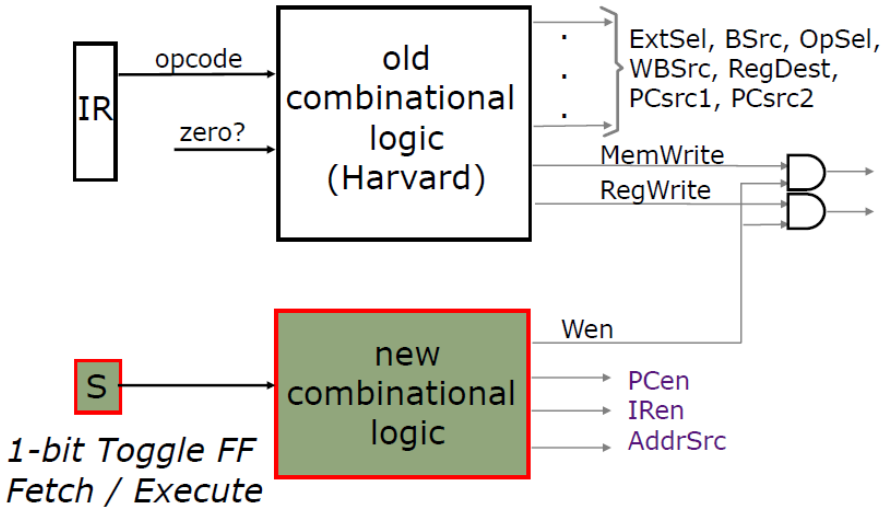
Ελεγκτής με Χρήση FSM 2 Καταστάσεων



- Υλοποίηση της FSM με χρήση ενός Flip-flop που κρατάει την κατάσταση

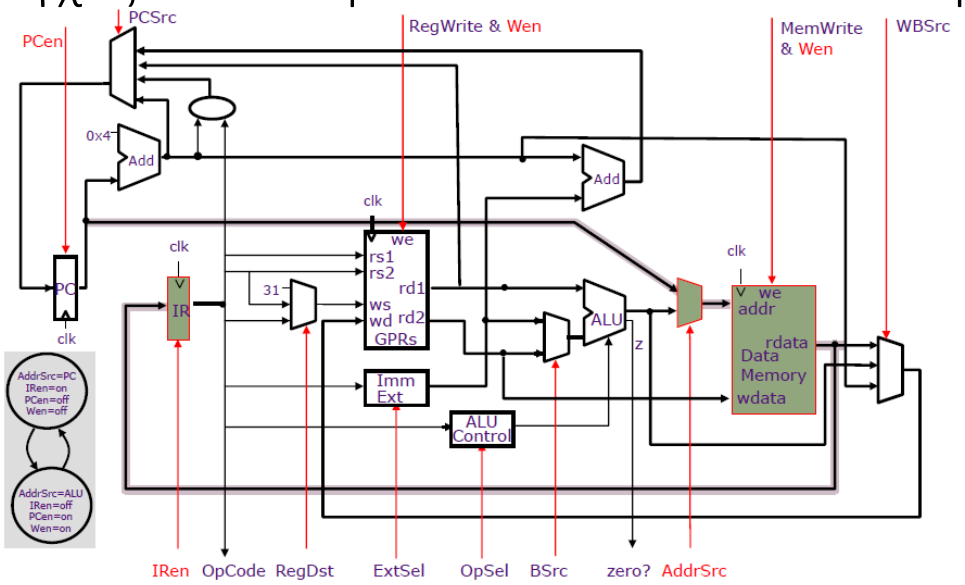
5

Λογικό Κύκλωμα Ελέγχου

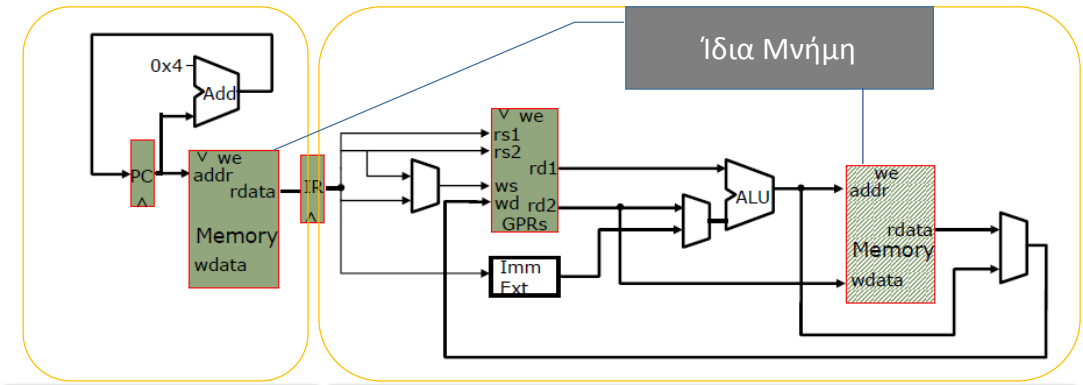


6

Έλεγχος και Datapath: 2 Κύκλοι ανά Εντολή!

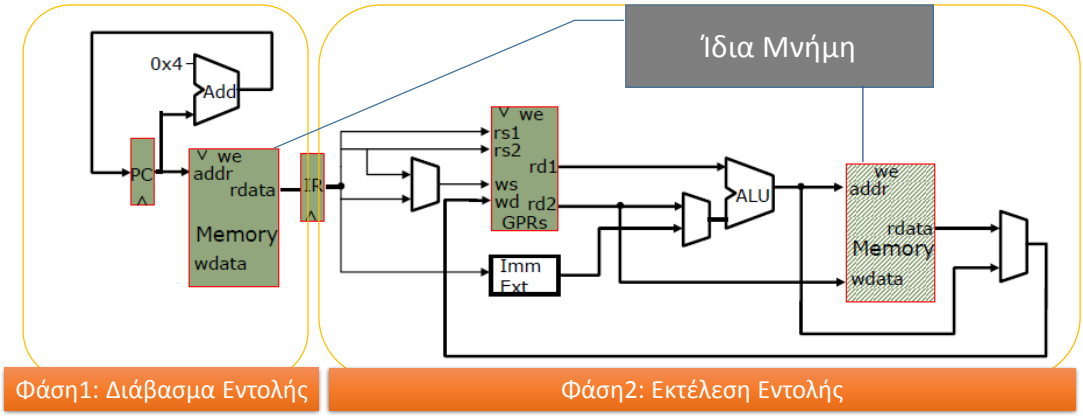


Datapath - Μικροαρχιτεκτονική



- Σε κάθε κύκλο ρολογιού μόνο MIA από τις φάσεις είναι ενεργή
- ⇒ Ένα μεγάλο κομμάτι του Datapath είναι αχρησιμοποίητο σε κάθε κύκλο ρολογιού

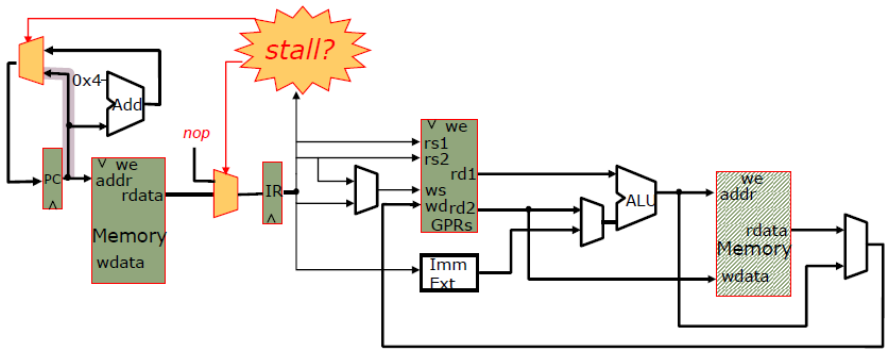
Επικάλυψη στην Εκτέλεση Εντολής



- Μπορούμε να έχουμε επικάλυψη των φάσεων ?
- ⇒ Ναι, εκτός αν ο καταχωρητής IR περιέχει Load ή Store εντολή
- Ποια φάση πρέπει να έχει προτεραιότητα? ⇒ Η εκτέλεση της εντολής
- Τι κάνουμε με τη Φάση1 ? ⇒ Καθυστέρηση, αλλά πώς?

9

Καθυστέρηση στην Προσκόμιση της Εντολής

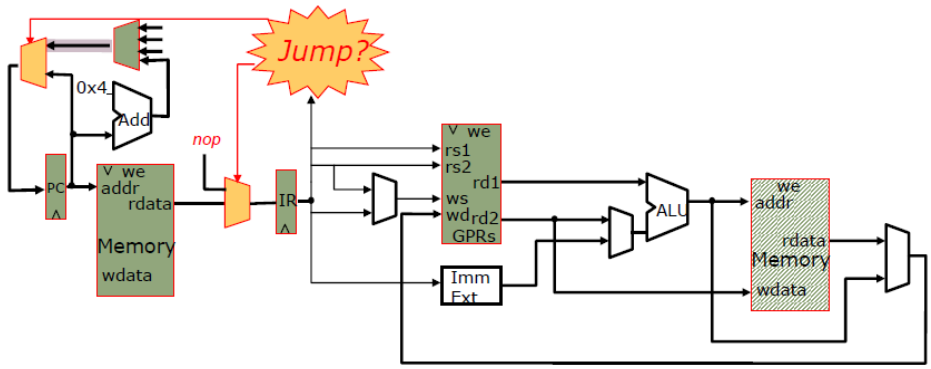


- Όταν ανιχνευθεί η περίπτωση για καθυστέρηση, τότε:
  - Μην φέρεις τη νέα εντολή και μην αυξήσεις τον PC
  - Βάλε ένα nop στον καταχωρητή IR
  - Φτιάξε την διεύθυνση μνήμης με την έξοδο της ALU ελέγχοντας τον πολυπλέκτη

Τι κάνουμε αν ο IR περιέχει εντολή jump ή branch ?

10

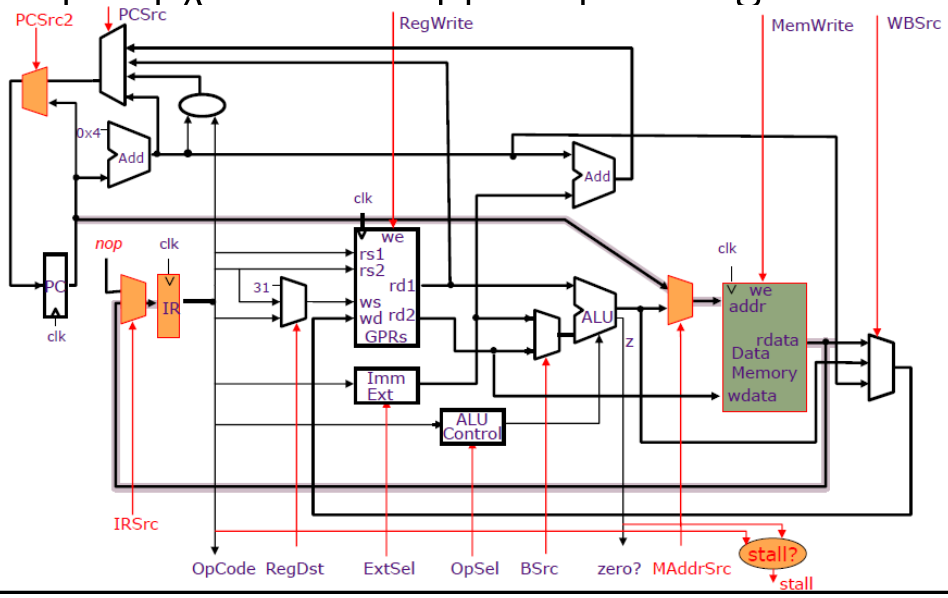
Καθυστέρηση λόγω Jump



- Όταν ο καταχωρητής IR περιέχει Jump ή επιτυχή Διακλάδωση, τότε:
  - Δεν έχουμε προβλήματα να ελέγξουμε την μνήμη, αλλά δεν υπάρχει ο σωστός PC
  - Βάλε ένα nop στον καταχωρητή IR
  - Εισαγωγή του σωστού nextPC (ή σωστή διεύθυνση διακλάδωσης) στον καταχωρητή PC

11

Μικροαρχιτεκτονική με Pipelining



12

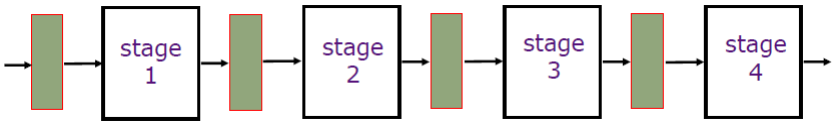
Πίνακας Ελέγχου

Opcode	Stall	Ext Sel	B Src	Op Sel	Mem W	Reg W	WB Src	Reg Dst	PC Src1	PC Src2	IR Src	MAddr Src
ALU	no	*	Reg	Func	no	yes	ALU	rd	pc+4	npc	mem	pc
ALUi	no	sE_{16}	Imm	Op	no	yes	ALU	rt	pc+4	npc	mem	pc
ALUiu	no	uE_{16}	Imm	Op	no	yes	ALU	rt	pc+4	npc	mem	pc
LW	yes	sE_{16}	Imm	+	no	yes	Mem	rt	pc+4	pc	nop	ALU
SW	yes	sE_{16}	Imm	+	yes	no	*	*	pc+4	pc	nop	ALU
$BEQZ_{z=1}$	yes	sE_{16}	*	0?	no	no	*	*	br	npc	nop	*
$BEQZ_{z=0}$	no	sE_{16}	*	0?	no	no	*	*	pc+4	npc	mem	pc
J	yes	*	*	*	no	no	*	*	jabs	npc	nop	*
JAL	yes	*	*	*	no	yes	PC	R31	jabs	npc	nop	*
JR	yes	*	*	*	no	no	*	*	rind	npc	nop	*
JALR	yes	*	*	*	no	yes	PC	R31	rind	npc	nop	*
NOP	no	*	*	*	no	no	*	*	pc+4	npc	mem	pc

BSrc = Reg / Imm ; WBSrc = ALU / Mem / PC; IRSrc = nop/mem; MAddSrc = pc/ALU
RegDst = rt / rd / R31; PCSrc1 = pc+4 / br / rind / jabs; PCSrc2 = pc/nPC

13

Ένα Ιδανικό Pipeline

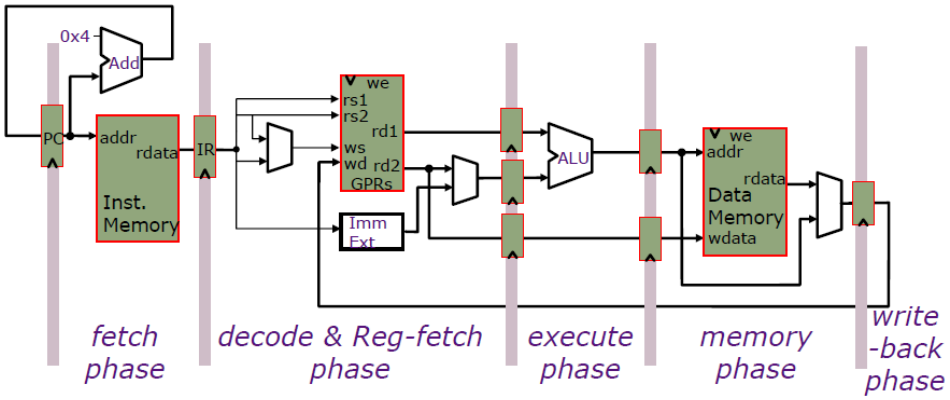


- Όλες οι εντολές (τμήματα επεξεργασμένης εντολής) περνάνε από όλα τα στάδια
- Δεν γίνεται διαμοιρασμός των πόρων από οποιαδήποτε στάδια
- Η καθυστέρηση διάδοσης/επεξεργασίας είναι η ίδια σε όλα τα στάδια
- Η είσοδος όποιου αντικειμένου σε ένα στάδιο δεν εξαρτάται από τα άλλα

- Τι πρέπει να γίνει σε ένα pipeline μικροαρχιτεκτονικής για την εκτέλεση εντολής ?

14

Datapath με Εκτέλεση Pipeline



- Η περίοδος του ρολογιού μπορεί να μειωθεί διαιρώντας την εκτέλεση της εντολής σε πολλαπλά στάδια

  $T_{clk} > \max \{t_{IM}, t_{RF}, t_{ALU}, t_{DM}, t_{RW}\}$ (= t_{DM} *πιθανά*)

- Εντούτοις, το CPI θα αυξηθεί εκτός και η εκτέλεση των εντολών γίνει με pipeline μορφή

15

Κατάτμηση του Datapath σε Στάδια

- Έστω ότι η μνήμη είναι σημαντικά πιο αργή από τα λοιπά στάδια

  t_{IM} = 10 units

  t_{DM} = 10 units

  t_{ALU} = 5 units

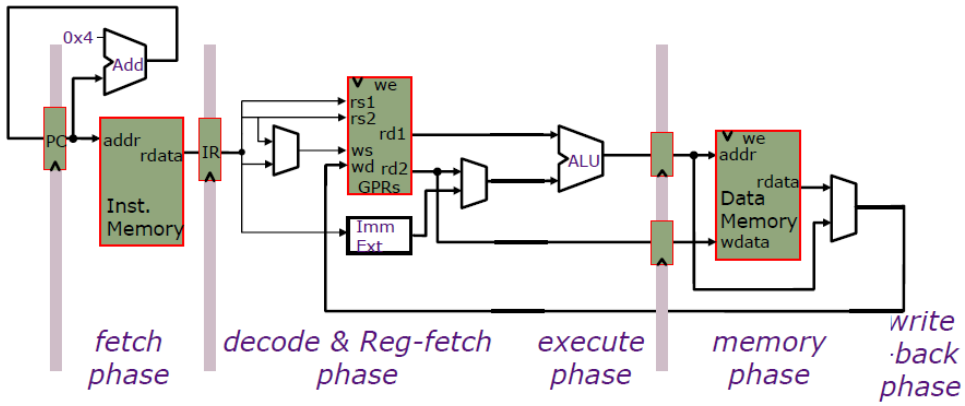
  t_{RF} = 1 unit

  t_{RW} = 1 unit

- Αφού το πιο αργό στάδιο καθορίζει την διάρκεια της περιόδου ρολογιού, ίσως είναι δυνατόν να γίνει συνδυασμός κάποιων σταδίων χωρία να μειωθεί η επίδοση

16

Pipelining: Πρώτη Προσέγγιση



- Tclk > max {tIM, tRF + tALU, tDM + tRW} = tDM + tRW
- Το στάδιο write-back διαρκεί πολύ λίγο και μπορεί να συνδυαστεί με το στάδιο της μνήμης

17

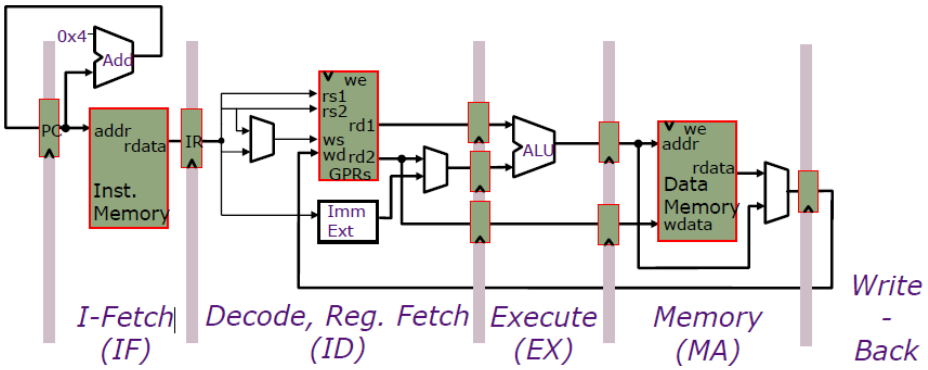
Pipelining: Μέγιστη Επιτάχυνση (Speedup)

Υποθέσεις	Χωρίς Pipelining (T clk)	Με Pipelining (T clk)	Speedup
1. $t_{IM} = t_{DM} = 10$, $t_{ALU} = 5$, $t_{RF} = t_{RW} = 1$ 4-stage pipeline	27	10	2.7
2. $t_{IM} = t_{DM} = t_{ALU} = t_{RF} = t_{RW} = 5$ 4-stage pipeline	25	10	2.5
3. $t_{IM} = t_{DM} = t_{ALU} = t_{RF} = t_{RW} = 5$ 5-stage pipeline	25	5	5.0

Μεγαλύτερη επιτάχυνση (Speedup) με περισσότερα στάδια

18

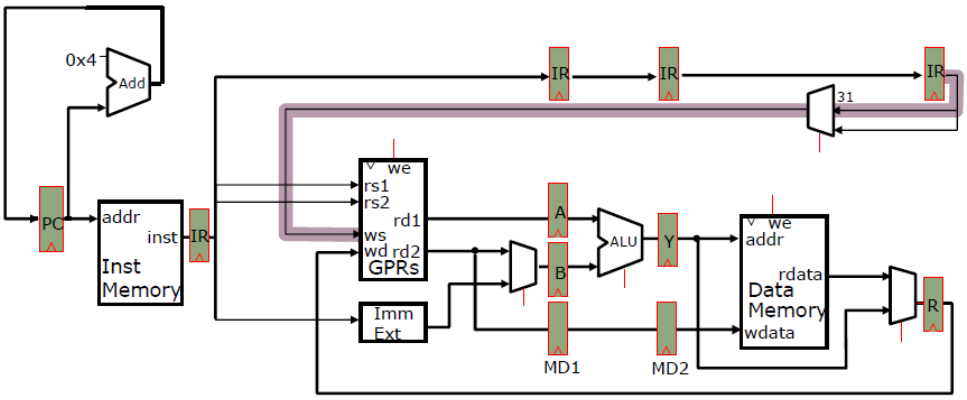
Εκτέλεση Εντολής με Pipeline 5 Σταδίων



Ροή Εκτέλεσης Εντολών

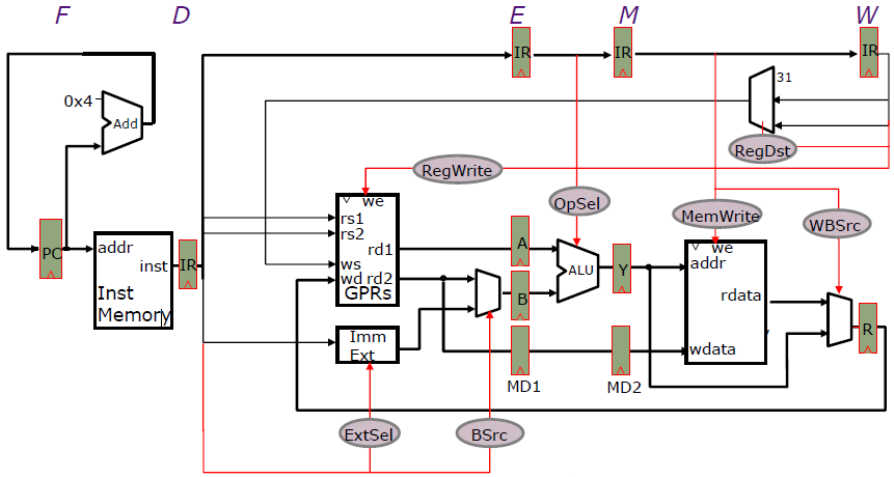
time	t0	t1	t2	t3	t4	t5	t6	t7	. . . .
instruction1	IF_1	ID_1	EX_1	MA_1	WB_1				
instruction2		IF_2	ID_2	EX_2	MA_2	WB_2			
instruction3			IF_3	ID_3	EX_3	MA_3	WB_3		
instruction4				IF_4	ID_4	EX_4	MA_4	WB_4	
instruction5					IF_5	ID_5	EX_5	MA_5	WB_5

Εκτέλεση με Pipelined Τρόπο



- Δεν είναι απόλυτα σωστό ⇒ απαιτείται ένας καταχωρητής εντολής (IR) για κάθε στάδιο

Datapath με Pipeline (χωρίς Jumps)



- Τι άλλο απαιτείται? ⇒ απαιτείται σωστός έλεγχος

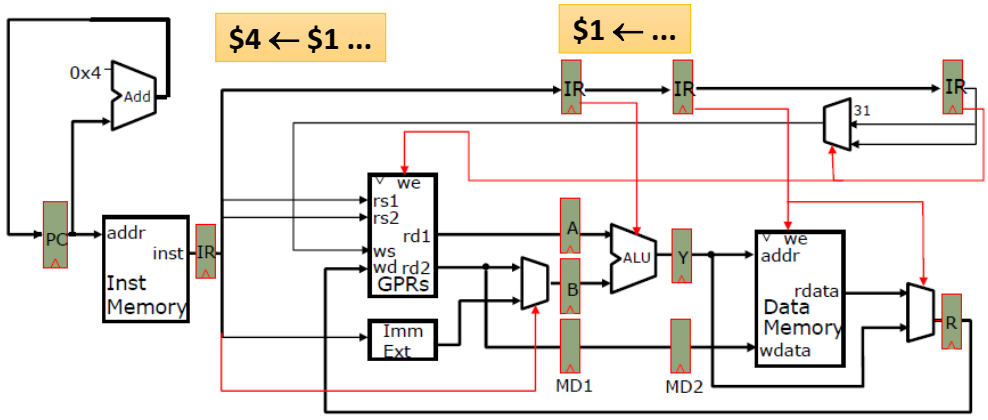
21

Ταυτόχρονη Εκτέλεση Εντολών σε Pipeline

- Μία εντολή μπορεί να χρειάζεται έναν πόρο που χρησιμοποιείται από άλλη εντολή που βρίσκεται υπό εκτέλεση και αυτή
  - ⇒ σύγκρουση πόρων
- Μία εντολή μπορεί να εξαρτάται από τιμή/αποτέλεσμα που παρήγαγε μία προηγούμενη εντολή
  - Η εξάρτηση μπορεί να οφείλεται σε δεδομένο
    ⇒ σύγκρουση δεδομένων
  - Η εξάρτηση μπορεί να οφείλεται σε υπολογισμό της επόμενης τιμής του PC
    ⇒ σύγκρουση ελέγχου

22

Εξάρτηση Δεδομένων

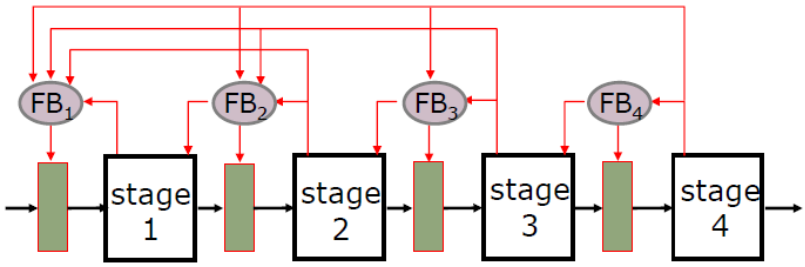


addi $1, $0, 10
addi $4, $1, 13

Επίλυση Εξάρτησης Δεδομένων

- Τρόπος #1: Αναμονή μέχρι το αποτέλεσμα να είναι διαθέσιμο, με πάγωμα των σταδίων pipeline (stall)
- Τρόπος #2: Προώθηση του αποτελέσματος μόλις υπολογισθεί στο κατάλληλο στάδιο του pipeline που το χρειάζεται (bypass)
- Τρόπος #3: Μάντεψε την εξάρτηση
  - Δύο περιπτώσεις:
  - Σωστή υπόθεση -> μην κάνεις τίποτα
  - Λάθος υπόθεση -> σταμάτα και ξανάρχισε την εκτέλεση

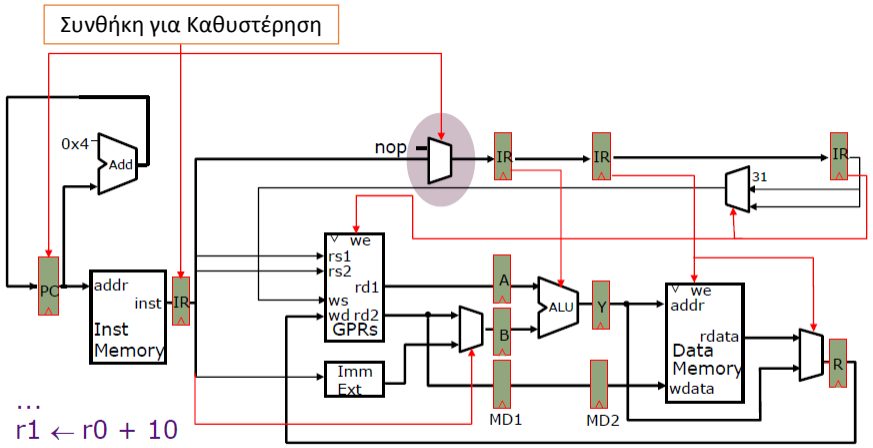
Τρόπος #1: Έλεγχος για Επίλυση Εξαρτήσεων



- Τα αργότερα στάδια παρέχουν πληροφορίες στα πρώιμα στάδια (feedback), ώστε ο έλεγχος να γνωρίζει αν πρέπει να ακυρώσει την εκτέλεση της εντολής
- Πρέπει ο έλεγχος να εγγυάται ότι το στάδιο i+1 θα ολοκληρωθεί και δεν επηρεάζεται από το στάδιο i (ειδάλως έχουμε deadlock)

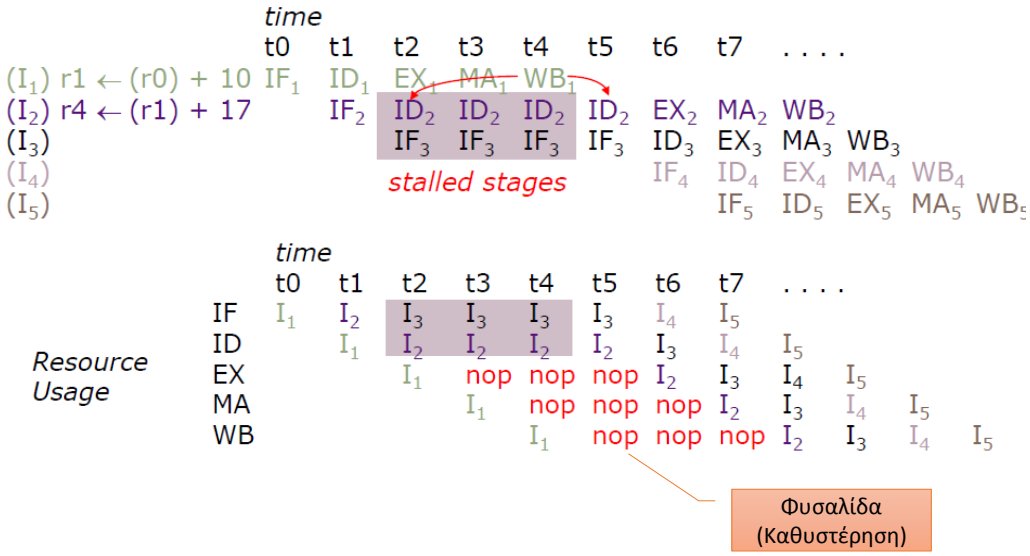
25

Τρόπος #1: Καθυστέρηση



26

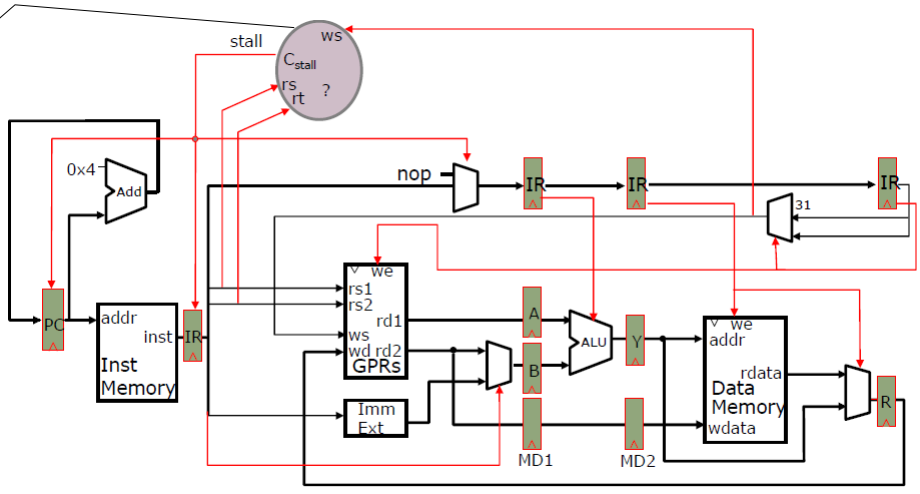
Στάδια με Καθυστέρηση



27

Λογική Ελέγχου Καθυστέρησης

Σύγκριση των καταχωρητών στο στάδιο αποκωδικοποίησης με τον καταχωρητή προορισμού στο στάδιο αποθήκευσης



28

Συνέχεια....

- Κίνδυνοι Ελέγχου (Control Hazards)
- Προσπέρασμα (Bypassing)
- Εκτέλεση με πρόβλεψη (Speculation)