Ο **επεξεργαστής**: Η **δίοδος δεδομένων** (**datapath**) και η **μονάδα ελέγχου** (**control**)

Σχεδίαση datapath

4 κατηγορίες εντολών:

Αριθμητικές-λογικές εντολές (add, sub, slt κλπ) –R Type

Εντολές αναφοράς στη μνήμη (lw, sw) – I Type

Εντολές διακλάδωσης (branch beq, bne) – I Type

Εντολές άλματος (jump j) – J Type

Τα σχήματα είναι από το βιβλίο COD2e/Patterson-Hennessy

cslab@ntua © 2006-07

…απλότητα στη σχεδίαση του ISA……

2 πρώτα βήματα κοινά σε κάθε εντολή (IF+ID):

– Στείλε το PC στη μνήμη (instruction memory) και φέρε (fetch) την εντολή (IF)

– Αποκωδικοποίησε την εντολή και διάβασε έναν ή δύο καταχωρητές-ορίσματα (ID-instruction decode+register file read)

Στη συνέχεια (EX):

–Οι arithmetic-logical χρησιμοποιούν την ALU για εκτέλεση της λειτουργίας του με βάση opcode και funct

–Οι memory-reference χρησιμοποιούν την ALU για υπολογισμό της τελικής δνσης του ορίσματος.

–Οι branch χρησιμοποιούν την ALU για σύγκριση

Κατόπιν (MEM-WB):

–Οι arithmetic-logical γράφουν το αποτέλεσμα της ALU πίσω σε ένα καταχωρητή του Register File

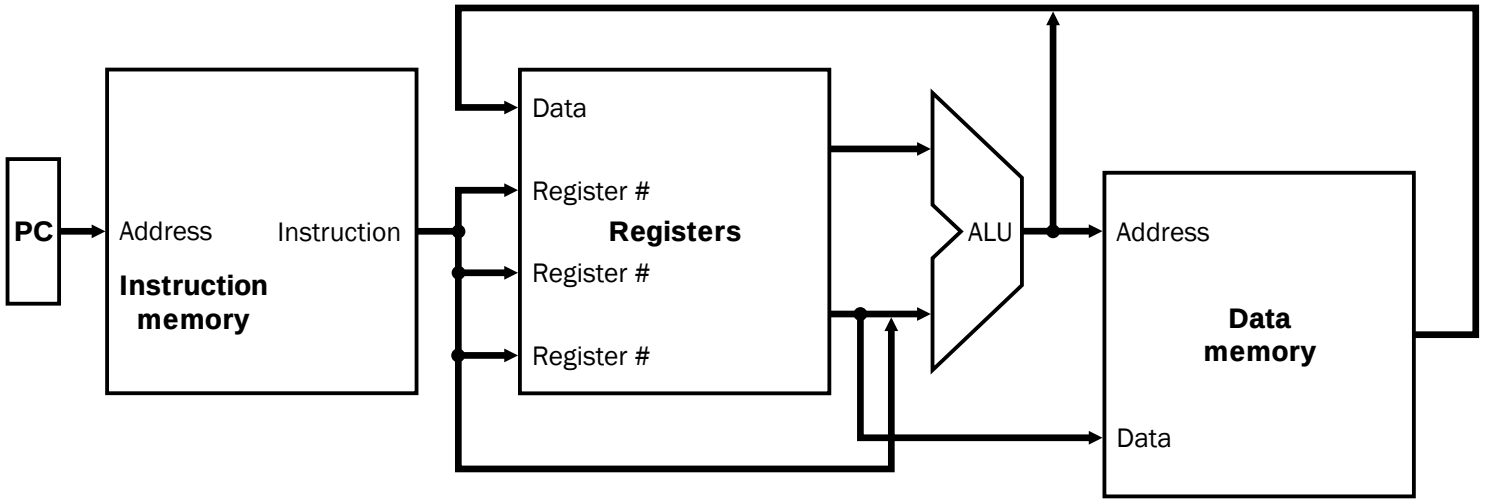
–Οι memory-reference διαβάζουν από τη μνήμη και γράφουν πίσω σε ένα καταχωρητή του Register File ή αποθηκεύουν στη μνήμη

–Οι branch αλλάζουν το περιεχόμενο του PC

IF-ID-EX-MEM-WB

cslab@ntua © 2006-07

Απλή (αφαιρετική) μορφή ενός datapath:

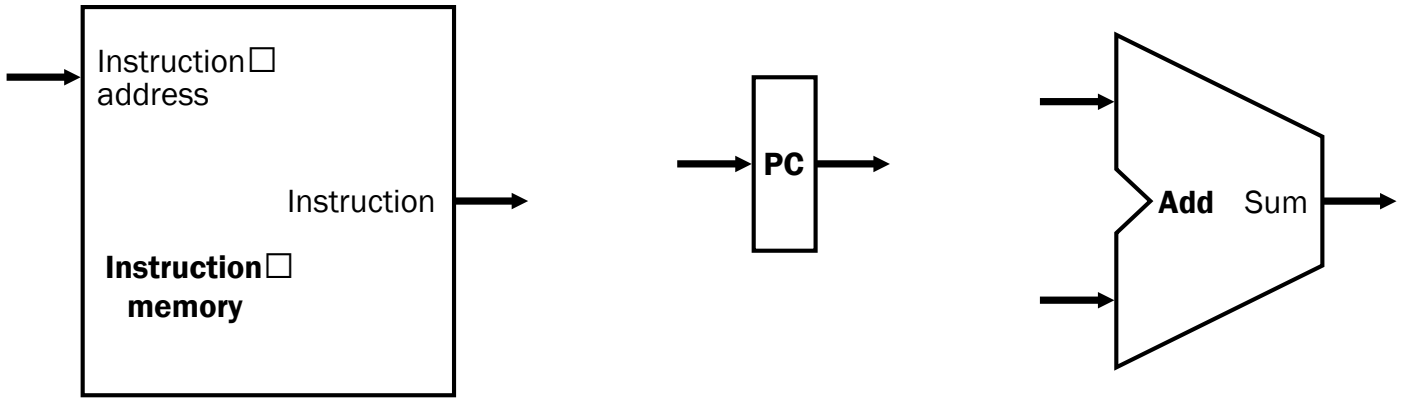


Αρχικά, θα κάνουμε σχεδίαση ενός κύκλου (single cycle)!

(*κάθε εντολή διαρκεί έναν κύκλο ρολογιού*)

cslab@ntua © 2006-07

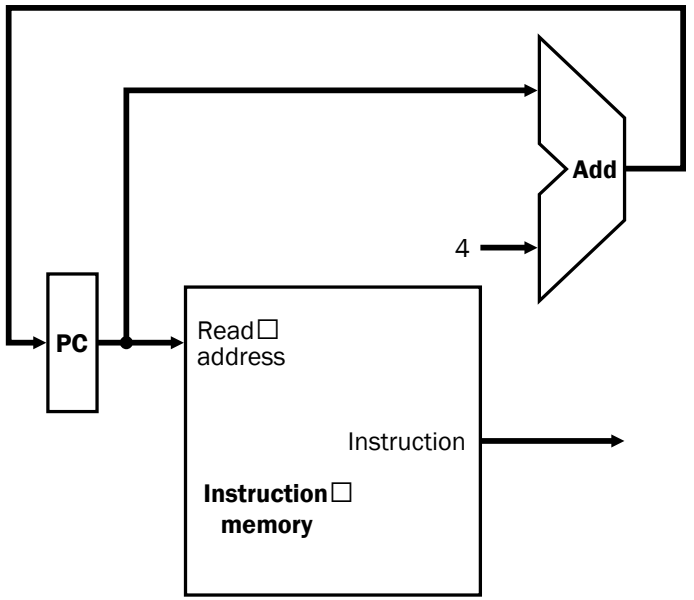
Στοιχεία για αποθήκευση και προσπέλαση εντολών:



a. Instruction memory b. Program counter c. Adder

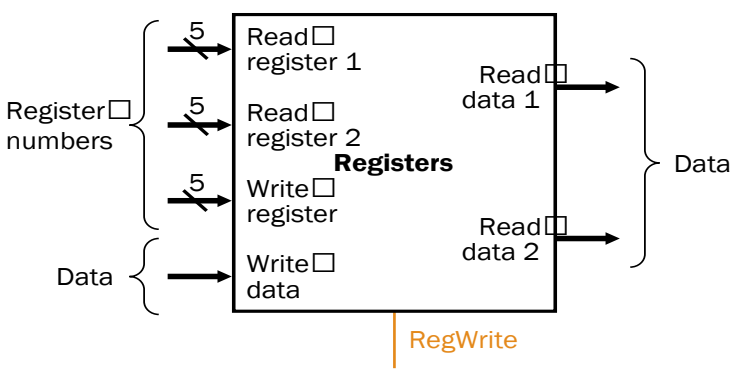
Αθροιστής για υπολογισμό PC+4 (επόμενη εντολή)

Fetching instructions and incrementing PC:

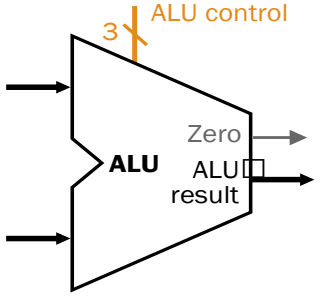


cslab@ntua © 2006-07

A. Υλοποίηση R-Type εντολών:



a. Registers



b. ALU

Register file:

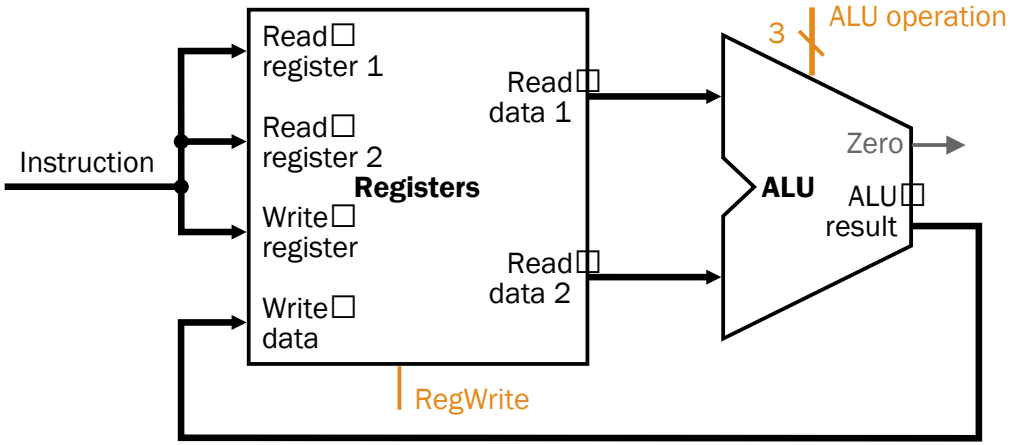
Two read ports, one write port (32 bit)
Δεν υπάρχει RegRead!
Υπάρχει RegWrite (edge triggered) όταν πρόκειται να γράψουμε Read and write Reg File στον ίδιο κύκλο (το read παίρνει την τιμή που γράφτηκε στον προηγούμενο κύκλο ενώ η τιμή που γράφεται είναι διαθέσιμη στον επόμενο κύκλο

ALU:

Zero output για branches

cslab@ntua © 2006-07

Τμήμα του Datapath για R-Type:



R-Type
(register type)

op	rs	rt	rd	shamt	funct
6 bits	*5bits*	*5bits*	*5bits*	*5bits*	*6bits*

`add $rd, $rs, $rt`

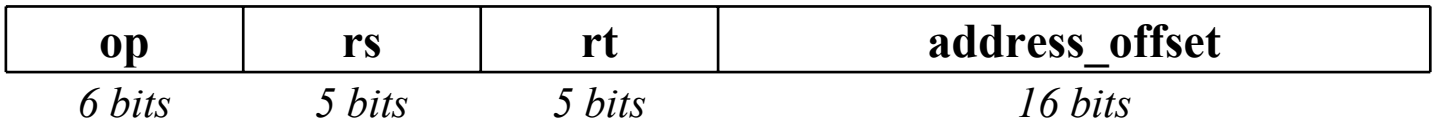
π.χ. add $s1, $s1, $s2

cslab@ntua © 2006-07

Σχεδίαση datapath για I-Type εντολές:

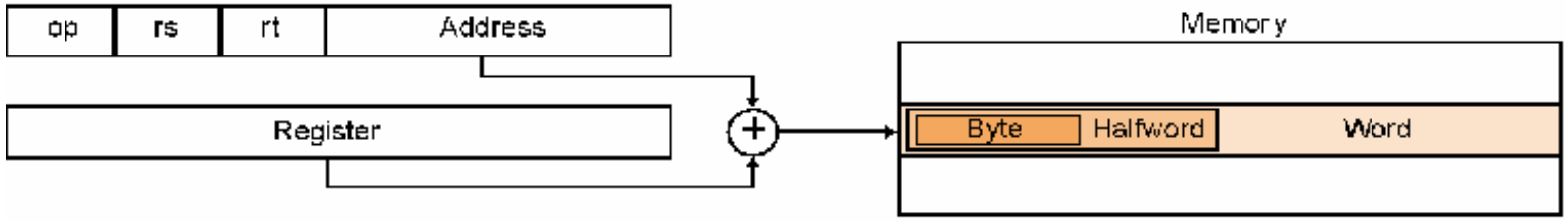
Πρώτα θα σχεδιάσουμε για load-stores

I-Type:




```
lw $rt, offset_value($rs)
                ή
sw $rt, offset_value($rs)
```

Offset_value: 16bit signed field
(Χρειάζεται sign extension γιατί προστίθεται σε 32 bit register)



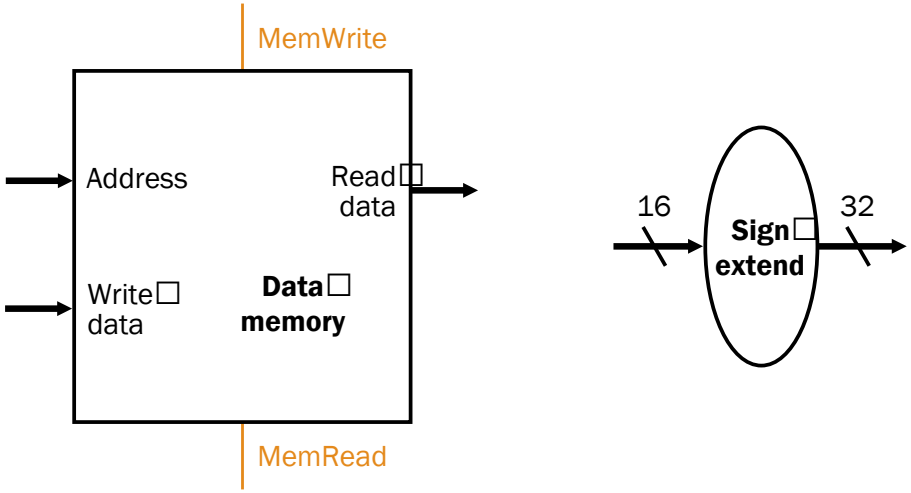
lw $rt, address($rs)

π.χ. lw $t1,100($s2)

Τα 3 πρώτα πεδία (op,rs, rt) έχουν το ίδιο όνομα και μέγεθος όπως και πριν

cslab@ntua © 2006-07

Επιπλέον δομικές μονάδες για load-store datapath:



a. Data memory unit

b. Sign-extension unit

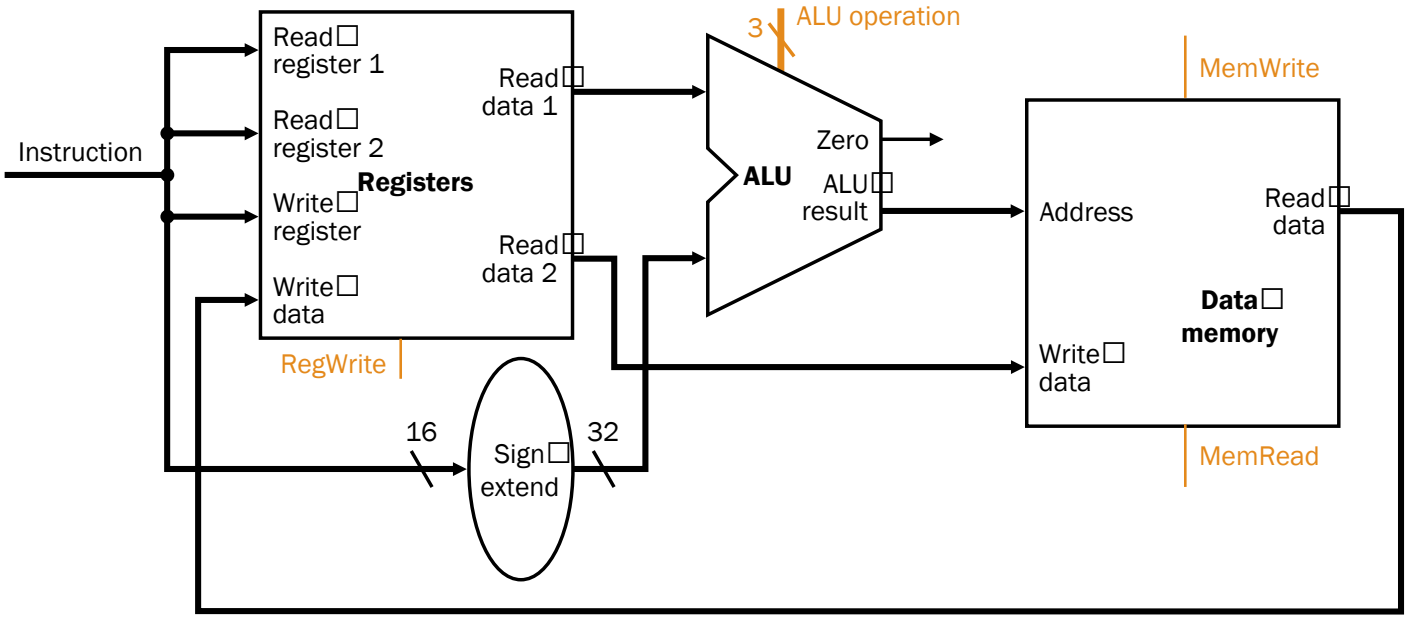
Memory:

address port ,read, write data port(32bit)

Sign extension!

cslab@ntua © 2006-07

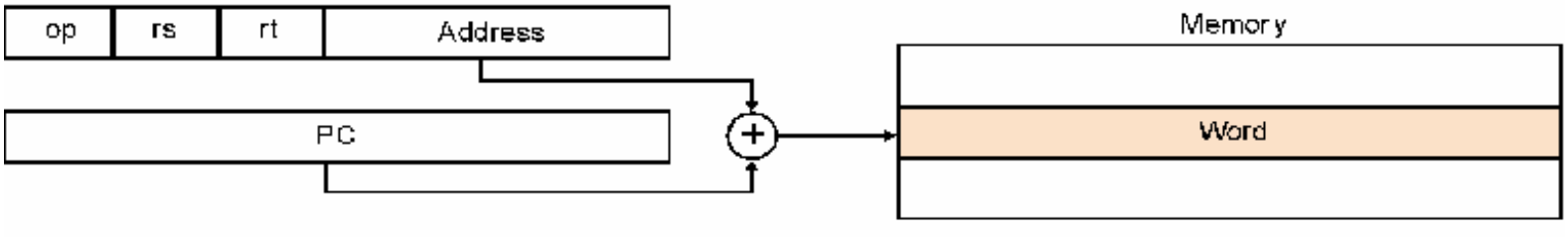
Datapath για load-store:



register file access → memory address calculation → read or write from memory → write back to rf if we have lw instruction

cslab@ntua © 2006-07

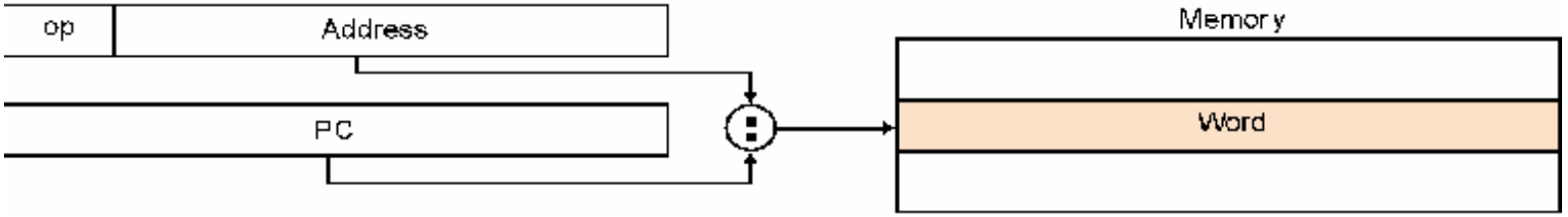
Σχεδίαση datapath για branch instructions:



bne $rs, $rt, address

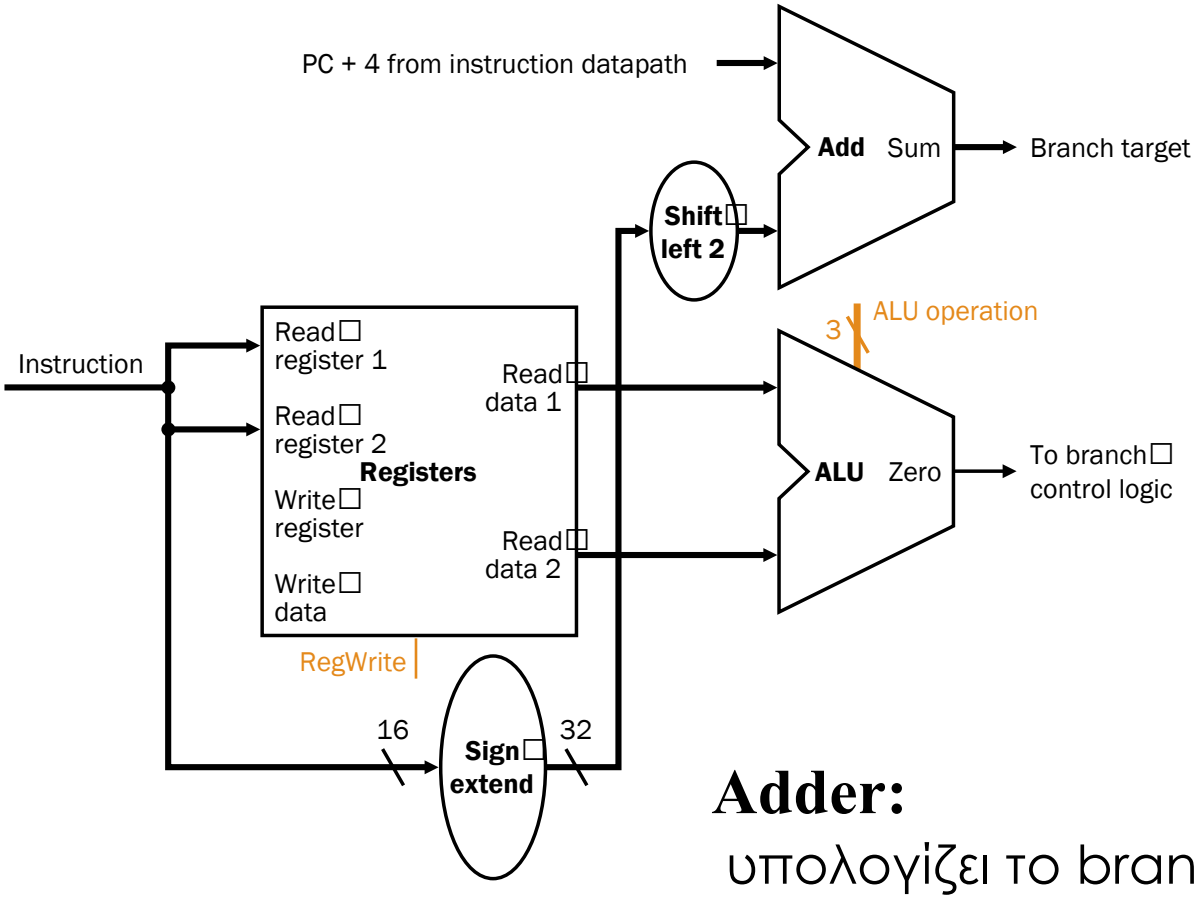
π.χ. bne $s0,$s1,L2

PC relative addressing άρα *address* χρειάζεται sign extension και x4 (αναφορά σε διευθύνσεις λέξεων)



cslab@ntua © 2006-07

Datapath για branch:



ALU:

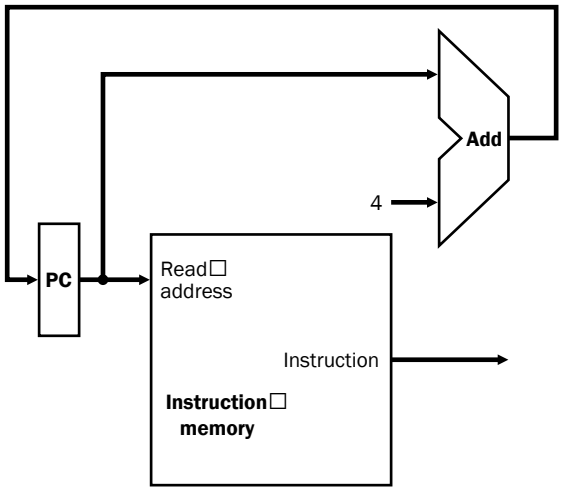
evaluates the branch condition

Adder:

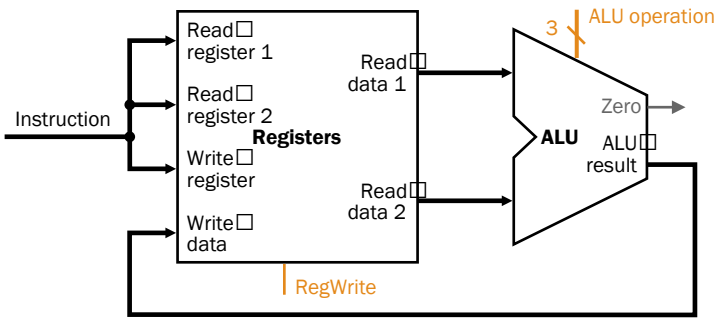
υπολογίζει το branch target σαν το άθροισμα του PC+4 και του sign extended, x4 όρισμα της εντολής

cslab@ntua © 2006-07

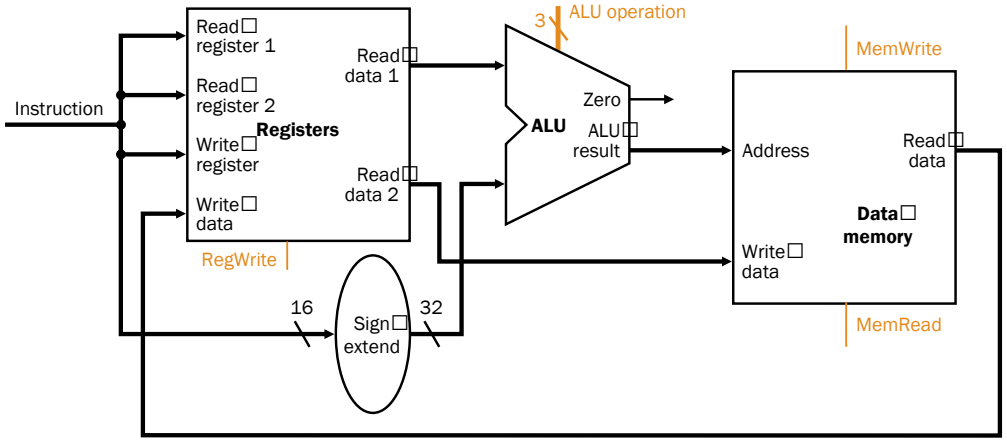
Ενώνοντας τα… όλα μαζί (δημιουργία single datapath):



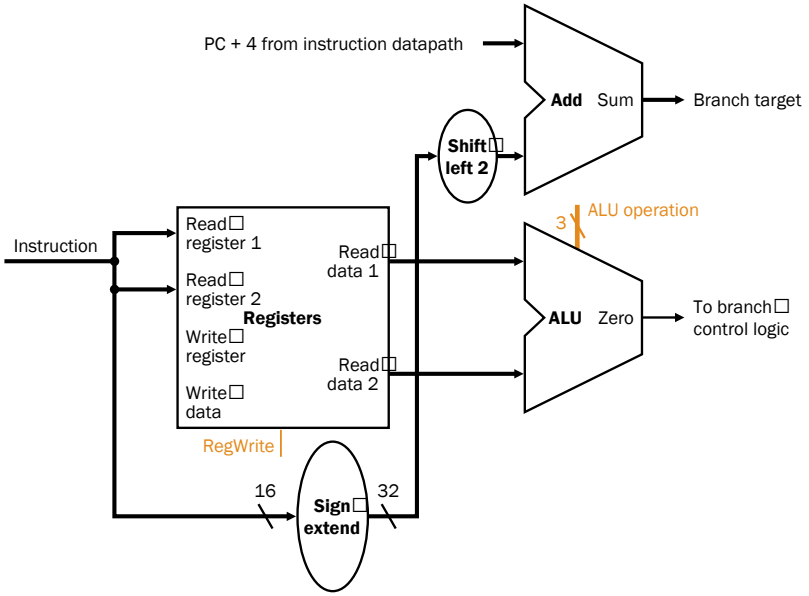
Fetch-Decode+Register File Read



R-Type



I-Type Load store

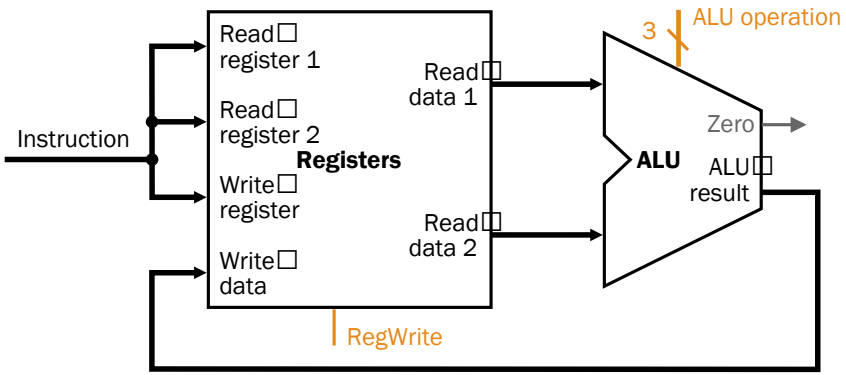


I-Type branch

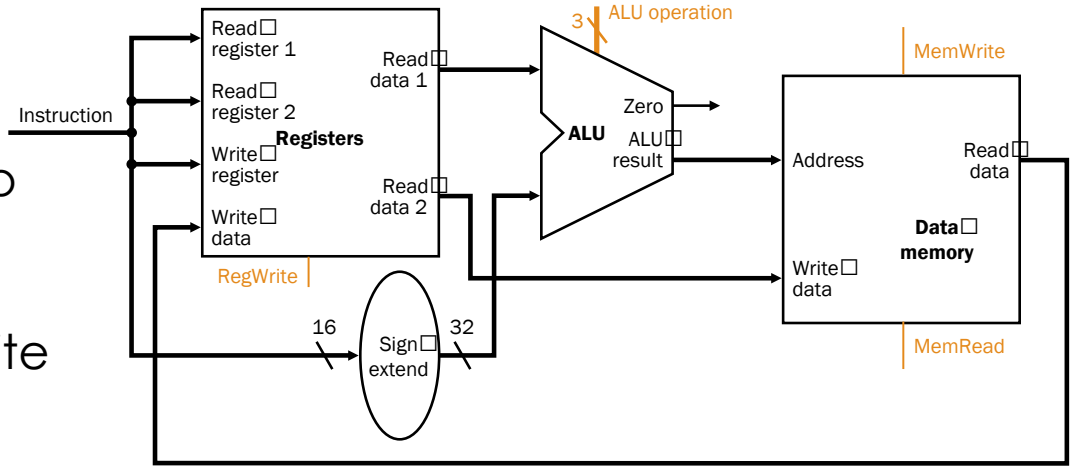
cslab@ntua © 2006-07

Μήπως ...μοιάζουν;

2η είσοδος στην ALU είναι είτε καταχωρητής (R-Type) είτε το sign-extended lower half (16bit) της εντολής (address_offset-αν είναι load-store



Η τιμή που αποθηκεύεται στο write data του register file έρχεται είτε από την ALU (R-Type) είτε από τη μνήμη (write back σε lw εντολή)

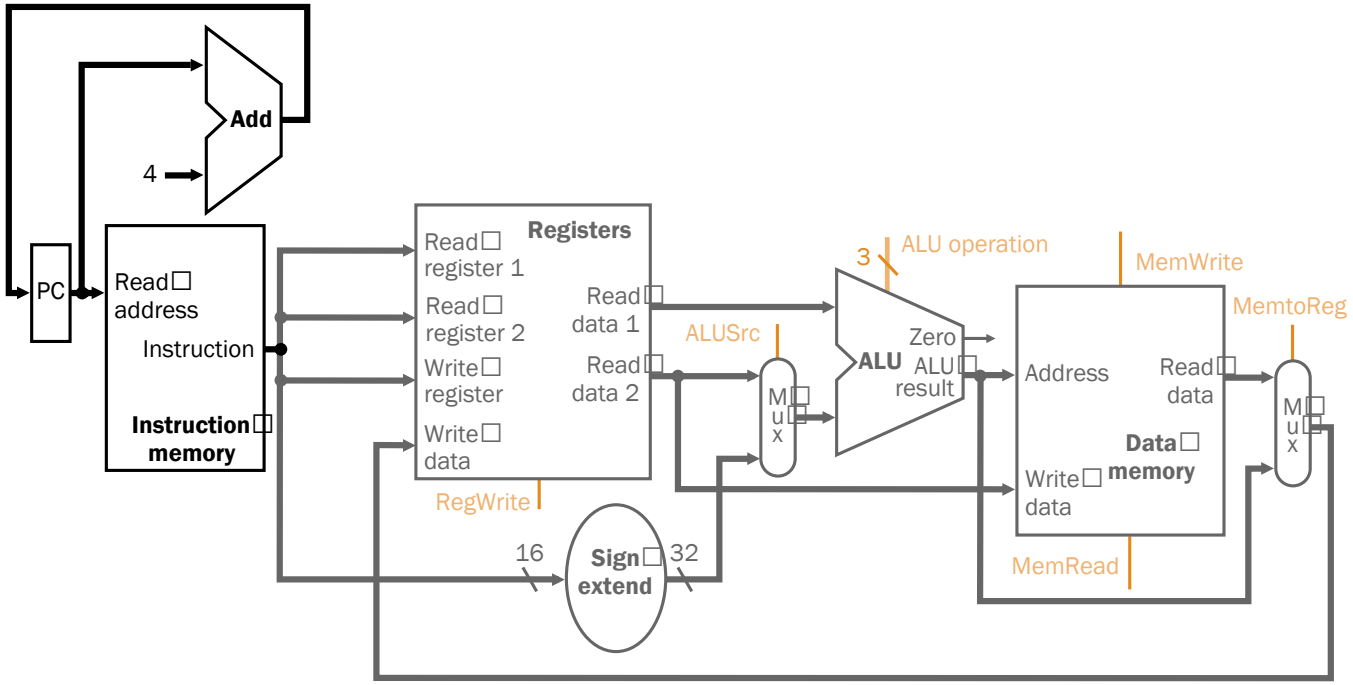


cslab@ntua © 2006-07

R-Type + load-store instructions combined datapath:

Χρησιμοποιούμε πολυπλέκτες (mux)

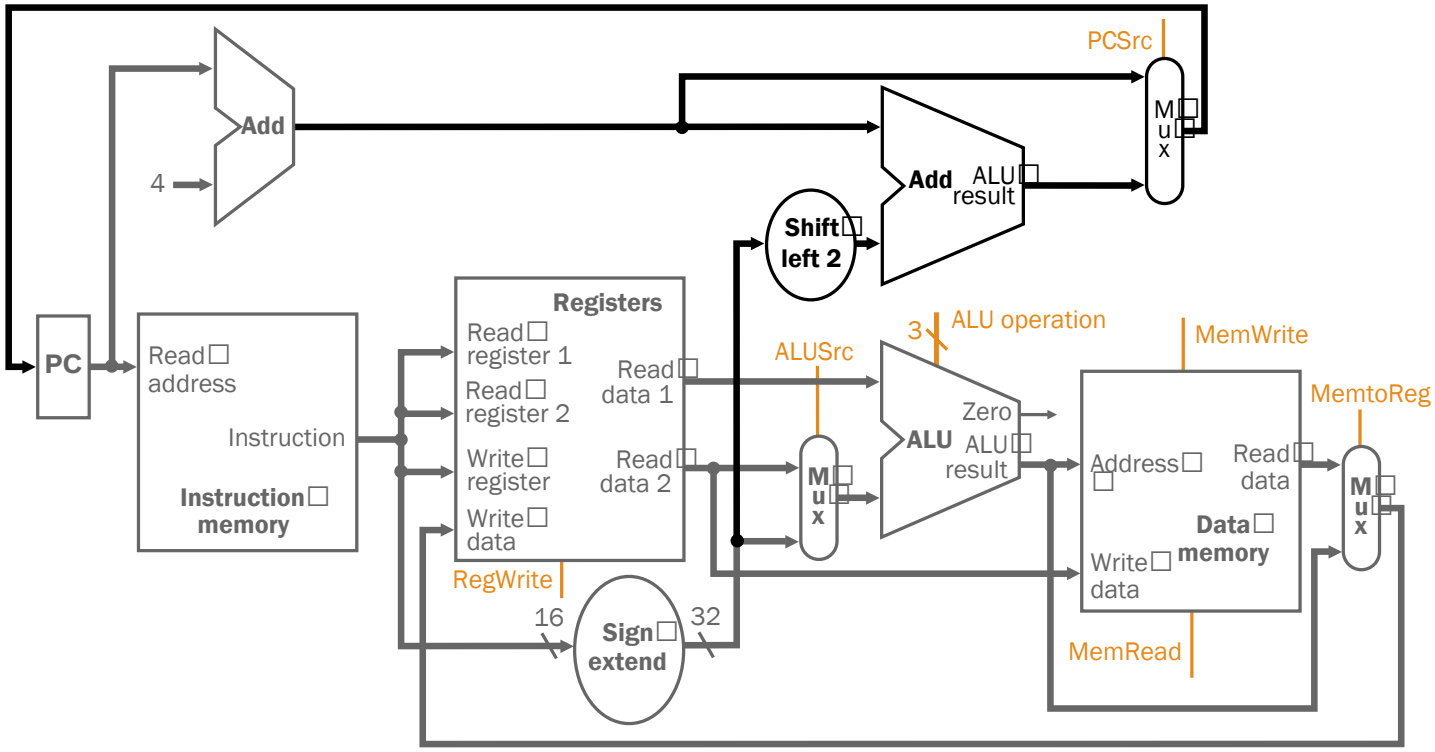
Single register file and single ALU:



Τι γίνεται με τα branch instructions;

cslab@ntua © 2006-07

Mux για επιλογή μεταξύ PC+4 και Label από branch instruction



cslab@ntua © 2006-07

Single-cyle υλοποίηση:

Διάρκεια κύκλου ίση με τη μεγαλύτερη εντολή-worst case delay (εδώ lw)

Αντιβαίνει με αρχή: Κάνε την πιο απλή περίπτωση γρήγορη.

Κάθε functional unit χρησιμοποιείται μια φορά σε κάθε κύκλο ανάγκη για πολλαπλό hardware.

Λύση: Multicycle υλοποίηση

Μικρότεροι κύκλοι ρολογιού, από τις καθυστερήσεις των επιμέρους functional units

cslab@ntua © 2006-07