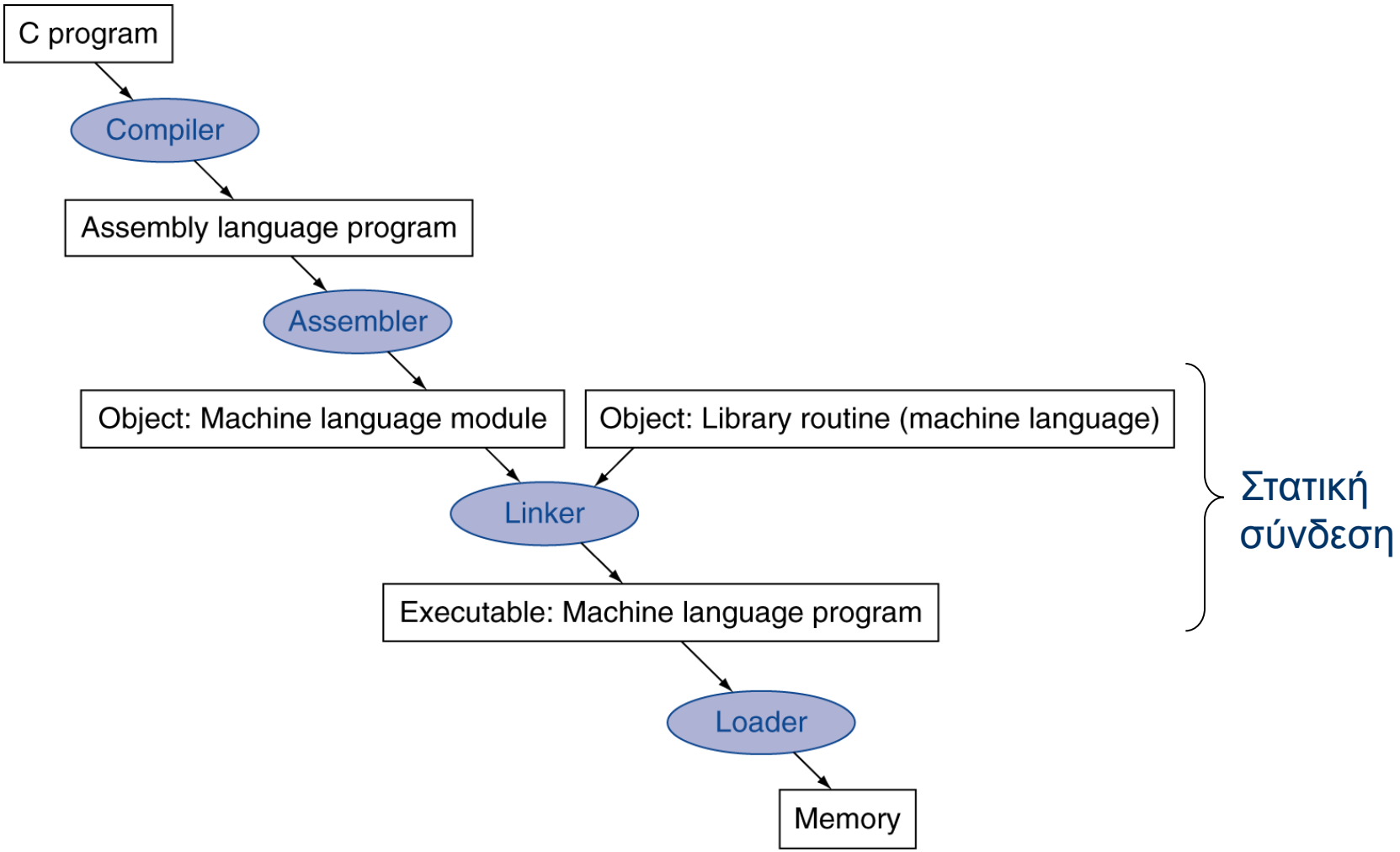
ΗΥ 134
Εισαγωγή στην Οργάνωση και στον Σχεδιασμό Υπολογιστών Ι

Διάλεξη 7
Μετάφραση ενός Προγράμματος
Εξαιρέσεις

Νίκος Μπέλλας
Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

Στάδια μετάφρασης ενός προγράμματος C



Εργασίες συμβολο-μεταφραστή (Assembler)

1. Μετάφραση ψευδοεντολών

- Οι περισσότερες εντολές του συμβολο-μεταφραστή έχουν 1-1 αντιστοιχία με εντολές της γλώσσας μηχανής.
- Ο συμβολο-μεταφραστής μετατρέπει τις ψευδοεντολές (pseudo-instructions) σε συνδιασμούς πραγματικών εντολών
- Οι ψευδοεντολές είναι:
  - Βοηθητικές εντολές οι οποίες αντικαθιστούν γνωστούς συνδυασμούς ή παραλλαγές εντολών της assembly που χρησιμοποιούνται για ειδικούς σκοπούς
  - 32-bit επεκτάσεις των σταθερών 16-bit στις εντολές I-type

```
move $t0, $t1      →  add $t0, $zero, $t1
blt $t0, $t1, L    →  slt $at, $t0, $t1
                      bne $at, $zero, L
li $s0,0x0123ABCD  →  lui $s0, 0x0123
                      ori $s0,$s0,0xABCD
```

  - $at (καταχωρητής 1): για χρήση απόκλειστικά από τον συμβολομεταφραστή

Εργασίες συμβολο-μεταφραστή

2. Παραγωγή αντικειμενικών υπομονάδων

- Ο μεταγλωττιστής ή ο συμβολομεταφραστής μετατρέπει το αρχικό πρόγραμμα σε ένα αντικειμενικό αρχείο (object file)
- Το αντικειμενικό αρχείο περιέχει
  - το πρόγραμμα μεταγλωττισμένο σε εντολές γλώσσας μηχανής
  - επιπλέον πληροφορίες και δεδομένα που χρειάζονται για την εκτέλεση του προγράμματος

Παραγωγή αντικειμενικής υπομονάδας

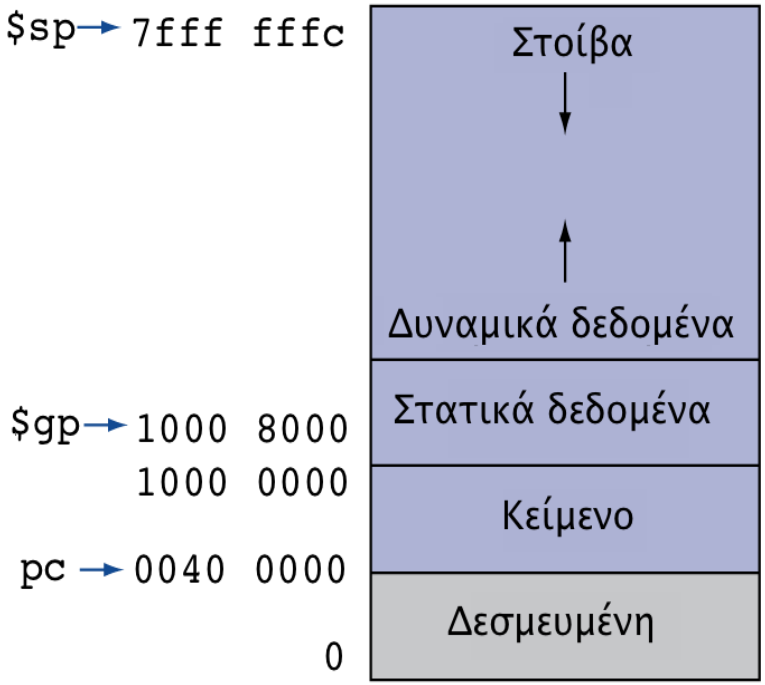
- Το αντικειμενικό αρχείο (*.o) περιέχει
  - **Επικεφαλίδα (Header):** περιγράφει το μέγεθος (σε bytes) και την θέση στην μνήμη των υπόλοιπων τμημάτων του αντικειμενικού αρχείου
  - **Τμήμα κειμένου (Text segment):** περιλαμβάνει τον κώδικα γλώσσας μηχανής
  - **Τμήμα στατικών δεδομένων (Static data segment):** Δεδομένα που διαρκούν καθ'όλη τη ζωή του προγράμματος (πχ. static variables)
  - **Πληροφορίες επανατοποθέτησης (Relocation info):** προσδιορίζουν εντολές και δεδομένα που εξαρτώνται από απόλυτες διευθύνσεις
  - **Πίνακας συμβόλων (Symbol table):** εξωτερικές αναφορές και συμβολικές αναφορές
  - **Πληροφορίες αποσφαλμάτωσης (Debug info):** για την αντιστοίχιση με τον πηγαίο κώδικα όταν κάνουμε debugging

Εργασίες προγράμματος σύνδεσης (linker)

- Σύνδεση αντικειμενικών υπομονάδων και παραγωγή του εκτελέσιμου αρχείου (executable) σε 3 στάδια:
  1. Τοποθέτηση τμημάτων κώδικα και δεδομένων συμβολικά στη μνήμη
  2. Προσδιορισμός διευθύνσεων ετικετών εντολών και δεδομένων
  3. Επιδιόρθωση (patching) εσωτερικών κι εξωτερικών αναφορών (internal and external references)
- Το πρόγραμμα σύνδεσης παράγει ένα εκτελέσιμο αρχείο (executable file)
  - Binary file : σειρά από 0 και 1
  - Μπορεί να εκτελεσθεί από υπολογιστή
- Ας δούμε ένα παράδειγμα:

Επανάληψη: Κατανομή μνήμης

- Κείμενο (Text): κώδικας
- Στατικά δεδομένα (*static data segment*): καθολικές μεταβλητές
  - πχ., στατικές μεταβλητές C, πίνακες, συμβολοσειρές
  - $gp : επιτρέπει εύκολη πρόσβαση στο τμήμα
- Δυναμικά δεδομένα: σωρός (heap)
  - π.χ., malloc στη C, new στη Java
- Τοπικές ματαβλητές: Στοίβα (Stack)



Παράδειγμα σύνδεσης προγραμμάτων

```
AA.c:
    int X;

    A () {
      ….
      lw $a0, X
      call B;
      ....
    }
BB.c:
    int Y;

    void B () {
      …
      sw $a1, Y
      call A;
      ....
    }
```

AA.o

Object file header			
	Name	Procedure A	
	Text size	100_{hex}	
	Data size	20_{hex}	
Text segment	Address	Instruction	
	0	`lw $a0, 0($gp)`	
	4	`jal 0`	
	…	…	
Data segment	0	(X)	
	…	…	
Relocation information	Address	Instruction type	Dependency
	0	`lw`	X
	4	`jal`	B
Symbol table	Label	Address	
	X	—	
	B	—	

BB.o

Object file header			
	Name	Procedure B	
	Text size	200_{hex}	
	Data size	30_{hex}	
Text segment	Address	Instruction	
	0	`sw $a1, 0($gp)`	
	4	`jal 0`	
	…	…	
Data segment	0	(Y)	
	…	…	
Relocation information	Address	Instruction type	Dependency
	0	`sw`	Y
	4	`jal`	A
Symbol table	Label	Address	
	Y	—	
	A	—	

- Θέλουμε να συνδέσουμε (link) τα δύο αντικειμενικά αρχεία (object files AA and BB)
- Οι εντολές φαίνονται εδώ σε συμβολική μορφή για να κατανοηθούν καλύτερα
- Στην πραγματικότητα, οι εντολές θα ήταν δυαδικοί αριθμοί

Παράδειγμα σύνδεσης προγραμμάτων

Executable file header		
	Text size	300_{hex}
	Data size	50_{hex}
Text segment	Address	Instruction
	$0040\ 0000_{hex}$	lw $a0, 8000_{hex}($gp)
	$0040\ 0004_{hex}$	jal 40 0100_{hex}
	...	...
	$0040\ 0100_{hex}$	sw $a1, 8020_{hex}($gp)
	$0040\ 0104_{hex}$	jal 40 0000_{hex}
	...	...
Data segment	Address	
	$1000\ 0000_{hex}$	(X)
	...	...
	$1000\ 0020_{hex}$	(Y)
	...	...

- Ο καταχωρητής $gp αρχικοποιείται πάντα με την τιμή 0x10008000
- Τελικό εκτελέσιμο αρχείο (executable file)

Φόρτωση προγράμματος (I)

- Αρχικά το εκτελέσιμο αρχείο βρίσκεται στον σκληρό δίσκο
- Ο φορτωτής (loader):
  1. Διαβάζει την επικεφαλίδα για να προσδιορίσει το μέγεθος των τμημάτων
  2. Δημιουργεί ένα χώρο διευθύνσεων στην κύρια μνήμη αρκετά μεγάλο για εντολές και δεδομένα
  3. Αντιγράφει εντολές και αρχικοποιημένα δεδομένα στη μνήμη (ο Loader είναι τμήμα του λειτουργικού συστήματος, Operating System)

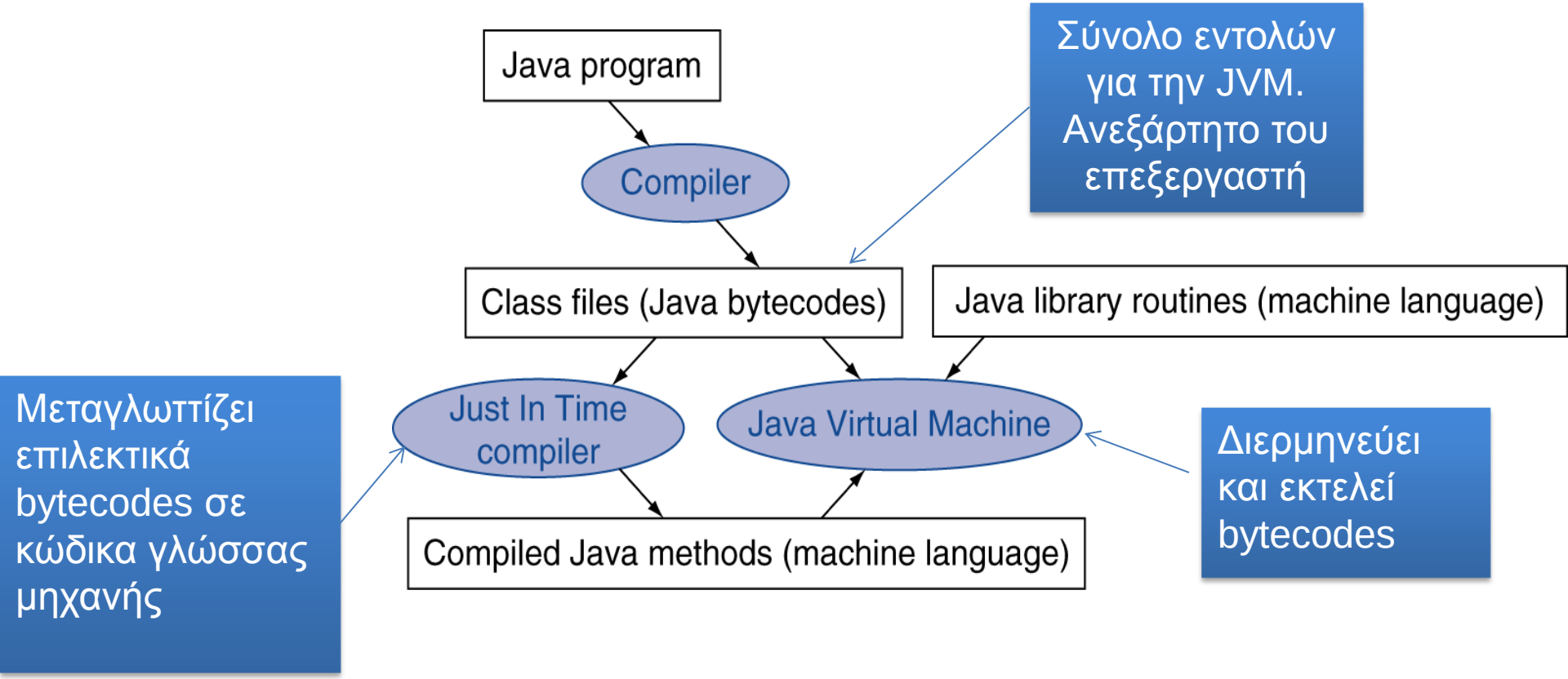
Φόρτωση προγράμματος (II)

- Φόρτωση από το δίσκο στη μνήμη:
  4. Αντιγράφει τις παραμέτρους του προγράμματος που δίνει ο χρήστης στη στοίβα (*argc, argv)*
  5. Αρχικοποιεί τους καταχωρητές (συμπεριλαμβανομένων των $sp, $fp, $gp) και θέτει τον $sp στην πρώτη ελεύθερη θέση
  6. Μεταπηδά στη ρουτίνα εκκίνησης
     - Η ρουτίνα αντιγράφει τις παραμέτρους στους $a0, … και καλεί τη main
     - Όταν η main επιστρέφει, κάνει κλήση συστήματος για έξοδο (exit syscall)

Προγράμματα σε Java

- Η γλώσσα προγραμματισμού Java έχει σχεδιαστεί με ένα διαφορετικό σύνολο στόχων από την C
- Στόχος ήταν η γρήγορη και ασφαλής εκτέλεση σε οποιονδήποτε υπολογιστή
  - Η ταχύτητα εκτέλεσης ενός προγράμματος ήταν (αρχικά) δευτερεύουσας σημασίας
- Η τυπική εκτέλεση ενός προγράμματος Java περιλαμβάνει δύο στάδια (stages)

Προγράμματα σε Java



Η τυπική σειρά εκτέλεσης είναι Compiler → Java Virtual Machine
Για υψηλότερη απόδοση, ένας Just In Time (JIT) compiler μπορεί να μεταγλωτίζει επιλεγμένα τμήματα του bytecode στην γλώσσα μηχανής του επεξεργαστή στον οποίο εκτελείται

Διακοπές και εξαιρέσεις (interrupts and exceptions)

- *Εξαιρέσεις* (exceptions): σφάλματα κατά την εκτέλεση εντολών του προγράμματος
  - Συνήθως προκαλούν την άμεση διακοπή του προγράμματος
  - Το γνωστό σε όλους *"segmentation fault"*
- Παραδείγματα εξαιρέσεων:
  - Υπερχείλιση σε αριθμητικές πράξεις
  - Διαίρεση με μηδέν
  - Άλμα στο χώρο *δεδομένων* από εντολές jr και jalr
  - Προσπέλαση του χώρου *προγράμματος* από εντολές lw και sw
- Ειδική περίπτωση εξαίρεσης: κλήσεις ρουτινών συστήματος (syscall)

Διακοπές και εξαιρέσεις (interrupts and exceptions)

- *Διακοπές* (interrupts): αιτήσεις για εξυπηρέτηση από συσκευές I/O
  - Για παράδειγμα, κάθε φορά που γράφετε από το πληκτρολόγιο
- Η διαχείριση των εξαιρέσεων και των διακοπών στον MIPS γίνεται από ένα ειδικό τμήμα του επεξεργαστή που ονομάζεται *συνεπεξεργαστής 0* (coprocessor 0)

Διακοπές και εξαιρέσεις

- Σημαντικότεροι καταχωρητές του συνεπεξεργαστή 0:

Register name	Register #	Usage
$badvaddr	8	address at which erroneous memory reference occurred
$status	12	interrupt enable and kernel/user bits
$cause	13	exception type and pending interrupts
$epc	14	address of instruction that caused exception (or that was executing when interrupt occurred)

- Εντολές μεταφοράς δεδομένων από και προς το συνεπεξεργαστή 0:

mfc0 $s0,$epc ⇔ $s0 = $epc

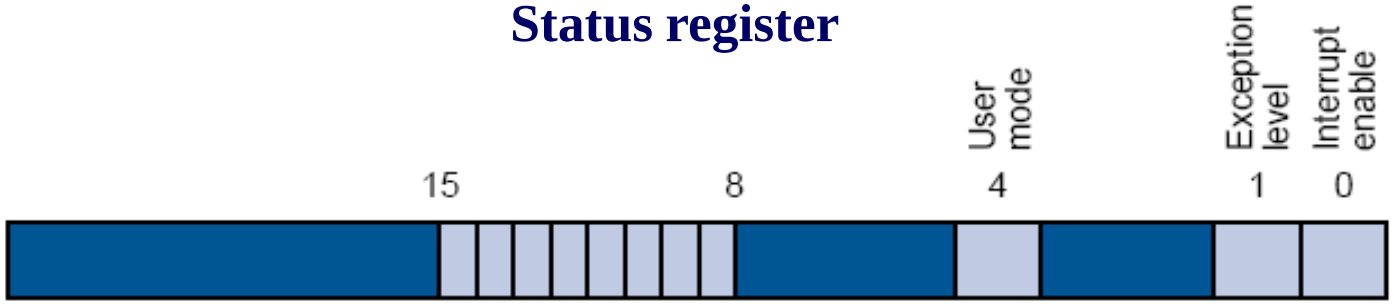
mtc0 $s0,$epc ⇔ $epc = $s0

lwc0 $epc,100($s1) ⇔ $epc = Mem[$s1+100]

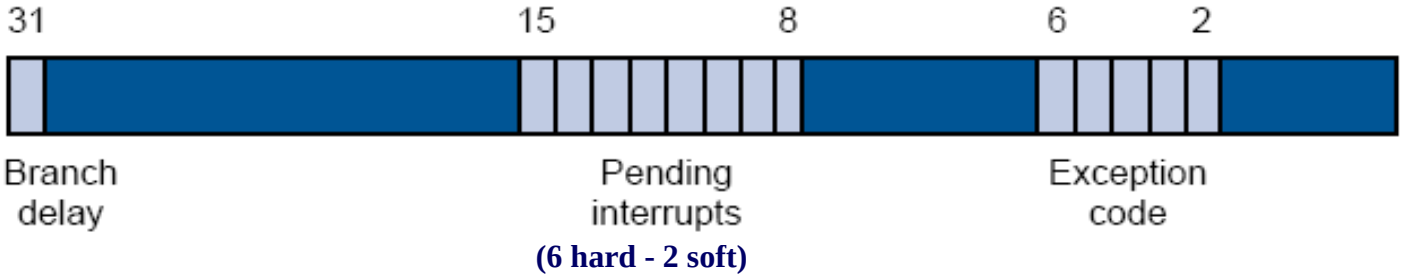
swc0 $epc,100($s1) ⇔ Mem[$s1+100] = $epc

Διακοπές και εξαιρέσεις

Status register



Cause register



Διακοπές και εξαιρέσεις

- Διακόπτουν την κανονική ροή εκτέλεσης του προγράμματος και εκτελούν άλμα σε μια διαδικασία *χειρισμού εξαίρεσης* (exception handler), η οποία βρίσκεται στο χώρο του *λειτουργικού συστήματος* (kernel) και όχι στο χώρο του χρήστη (user)
- Βήματα που εκτελούνται από το ***hardware*** όταν προκληθεί μια εξαίρεση/διακοπή:
  - Αποθήκευση του PC στον καταχωρητή $epc (ενδεχομένως και της λανθασμένης διεύθυνσης αναφοράς στον καταχωρητή $badvaddr). Για παράδειγμα, εάν η εντολή lw $s0, 0($sp) δημιούργησε μιά εξαίρεση, τότε $badvaddr ← 0($sp)
  - Καταγραφή του κωδικού εξαίρεσης στον καταχωρητή $cause
  - *Γενική απενεργοποίηση* των διακοπών και ανάθεση kernel/user λειτουργίας σε *kernel* mode στον καταχωρητή $status
  - Άλμα στη δεκαεξαδική διεύθυνση 0x80000180 όπου βρίσκεται ο exception handler

Διακοπές και εξαιρέσεις

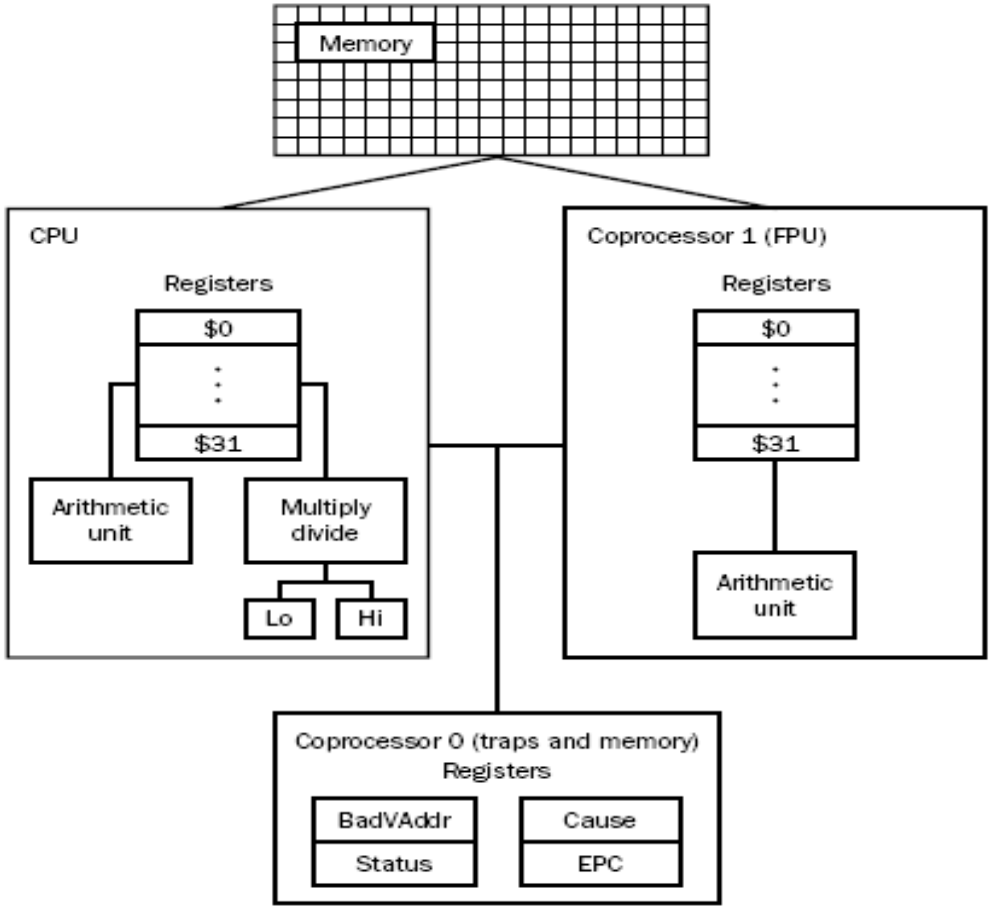
- Βήματα που εκτελούνται από το ***λειτουργικό σύστημα*** όταν προκληθεί μια εξαίρεση/διακοπή:
  - Ανάγνωση του καταχωρητή $cause για προσδιορισμό του κωδικού εξαίρεσης
  - Άλμα σε μια ρουτίνα του λειτουργικού συστήματος ανάλογα με τον κωδικό εξαίρεσης (πιθανόν με τη βοήθεια ενός jump address table)
  - Χειρισμός εξαίρεσης (ή εξυπηρέτηση της συσκευής που προκάλεσε διακοπή)
  - Επιστροφή στη ρουτίνα του exception handler
  - Εντολή *eret* (exception return) για επιστροφή στο πρόγραμμα του χρήστη - εφόσον δεν χρειάστηκε αυτό να τερματίσει κατά το χειρισμό της εξαίρεσης - η οποία βασικά εκτελεί:

    mfc0 $k0,$epc

    addi $k0,$k0,4

    jr $k0
- Εντολή *break n* προκαλεί εξαίρεση με τον κωδικό n

Γενική δομή του επεξεργαστή MIPS



Διακοπές και εξαιρέσεις

- Σημαντικότεροι κωδικοί διακοπών/εξαιρέσεων του MIPS:

Number	Name	Description
0	Int	external interrupt
4	AdEL	*address load error (instruction or data)*
5	AdES	address store error (data only)
6	IBE	bus error on instruction load
7	DBE	bus error on data load or store
8	Sys	syscall exception
9	Bp	breakpoint exception
10	RI	reserved (illegal) instruction
11	CpU	coprocessor unimplemented
12	Ov	arithmetic overflow
13	Tr	trap (general) exception
15	FPE	floating point exception