ΗΥ 232
Οργάνωση και Σχεδίαση Υπολογιστών

Διάλεξη 13

Εξαιρέσεις και Διακοπές (Exceptions and Interrupts)

Νίκος Μπέλλας
Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

Γιατί χρειάζονται interrupts σε ένα υπολογιστικό σύστημα;

- Τα μοντέρνα λειτουργικά συστήματα (Operating Systems, OS) χρονο-προγραμματίζουν (schedule) διεργασίες χρησιμοποιώντας *pre-emption*:
- Κάθε φορά που ένας υπολογιστής θέλει να τρέξει δύο προγράμματα Α και Β, το λειτουργικό σύστημα περιοδικά μεταφέρει τον έλεγχο από το ένα στο άλλο. Πιο συγκεκριμένα, το λειτουργικό:
    - Σταματάει το πρόγραμμα Α
    - Μεταφέρει τα δεδομένα των καταχωρητών στην κύρια μνήμη
    - Μεταφέρει τα δεδομένα του Β από την κύρια μνήμη στους καταχωρητές
    - Ξεκινάει τον πρόγραμμα Β

Πως σταματάει ένα λειτουργικό σύστημα το πρόγραμμα Α;

Απάντηση: Χρησιμοποιεί ένα *interrupt*.

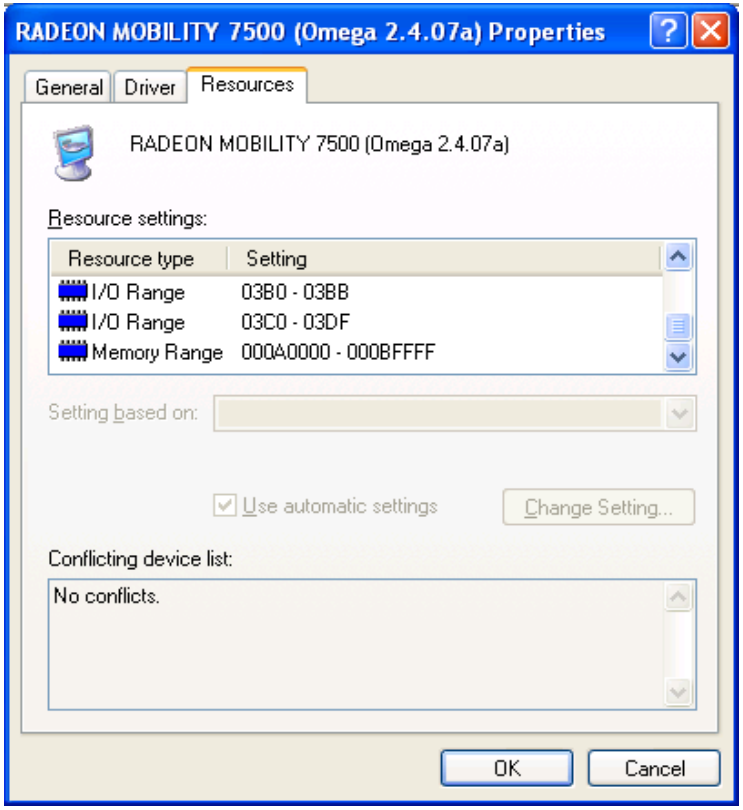
Προγραμματισμός I/O, Interrupts, Exceptions (Διακοπές και Εξαιρέσεις)

- Οι περισσότερες αιτήσεις για I/O γίνονται από το λογισμικό για να γίνουν μεταφορές δεδομένων μεταξύ I/O περιφερειακών και της κύριας μνήμης

- Οι δύο πιο συνηθισμένοι τρόποι επικοινωνίας με εξωτερικές συσκευές είναι:
  - Memory-mapped I/O
  - Isolated I/O
- Οι πιο συνηθισμένοι μέθοδοι για να προγραμματίσουμε την μεταφορά δεδομένων μεταξύ κύριας μνήμης και I/O περιφερειακών:
  - Interrupt-driven I/O
  - Programmed I/O
- Η χρήση Interrupt-driven I/O δημιουργεί την ανάγκη να συζητήσουμε:
  - Διακοπές (Interrupts)
  - Εξαιρέσεις (Exceptions)

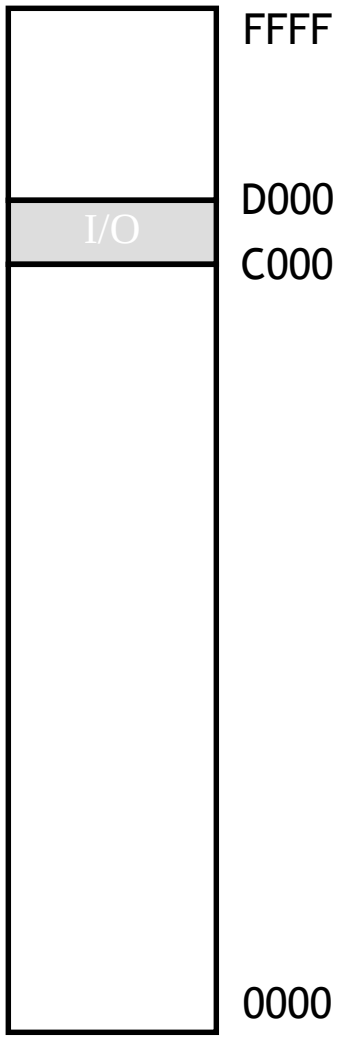
Επικοινωνία με περιφερειακές συσκευές

- Περιφερειακές συσκευές όπως το πληκτρολόγιο, ποντίκι, οθόνη, κάρτα δικτύου κλπ. θεωρούνται από τον επεξεργαστή σαν μνήμες.
  - Ο επεξεργαστής μπορεί να τις διαβάσει ή να γράψει σε αυτές όπως ακριβώς κάνει και σε οποιαδήποτε μνήμη
- Στο παράδειγμα, η κάρτα γραφικών μπορεί να προσπελαστεί μέσω των διευθύνσεων 3B0-3BB, 3C0-3DF και A0000-BFFFF.
- Αυτές οι διευθύνσεις μπορούν να προσπελαστούν με δύο τρόπους:
  - Memory-mapped I/O
  - Isolated I/O

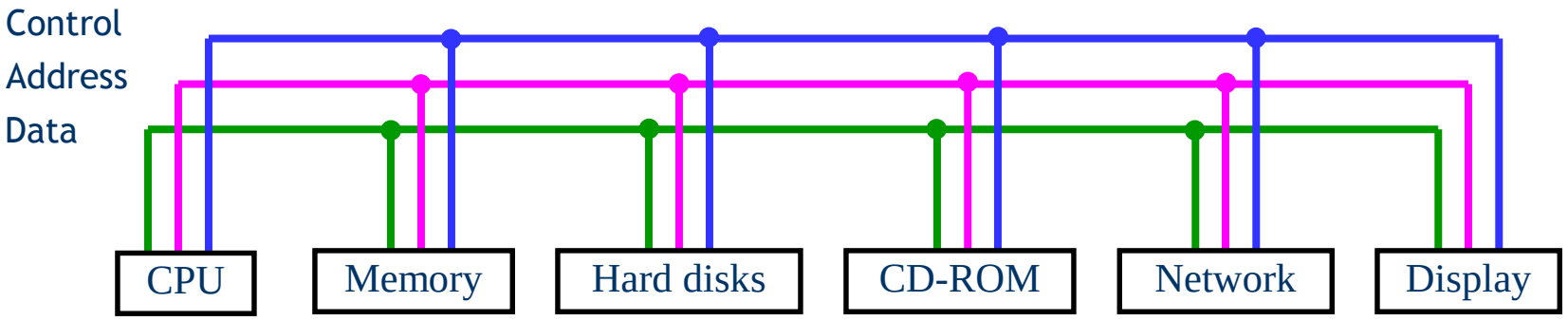


Memory-mapped I/O

- Σύμφωνα με το μοντέλο memory-mapped I/O, ο χώρος διευθύνσεων της μνήμης διαιρείται σε 2 τμήματα:
  - Ένα τμήμα που αναφέρεται στην φυσική κύρια μνήμη
  - Ένα τμήμα που αναφέρεται σε περιφερειακές συσκευές.
- Για παράδειγμα, ο παλιός *Apple IIe* είχε 16-bit χώρο διευθύνσεων και μπορούσε να προσπελάσει 64 KB κύριας μνήμης.
  - Οι διευθύνσεις *C000-CFFF* χρησιμοποιούνταν για να προσπελασθούν οι περιφερειακές συσκευές.
  - Η διεύθυνση *C010* αναφέρεται στο πληκτρολόγιο ενώ η *C030* αναφέρεται στο ηχείο.
  - Μερικές σε περιφερειακές συσκευές χρειάζονται πολλαπλές θέσεις μνήμης



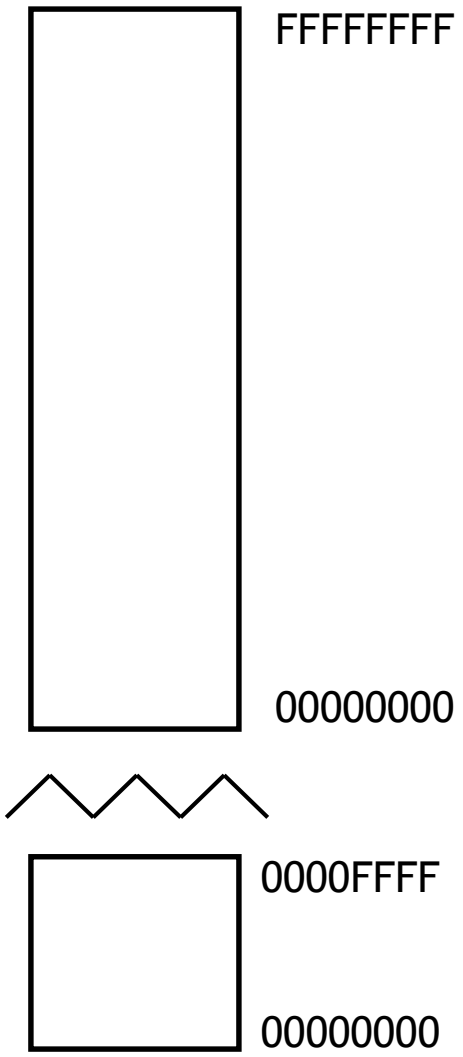
Προγραμματισμός memory-mapped I/O



- Για να γράψει σε μια περιφερειακή συσκευή, η CPU στέλνει πρώτα την διεύθυνση που αντιστοιχεί στην συσκευή στον Δίαυλο Διευθύνσεων (Address Bus). Ταυτόχρονα, στέλνει και τα δεδομένα στο Δίαυλο Δεδομένων (Data Bus). Για παράδειγμα, για να γράψουμε *0xABAB* στο ηχείο, η CPU θέτει Address Bus = *0xC030* , Data Bus = *0xABAB*
- Από την πλευρά της, κάθε περιφερειακή συσκευή παρακολουθεί το Address Bus για να διαπιστώσει εάν είναι ο παραλήπτης μιας προσπέλασης.
  - Το ηχείο αποκρίνεται σε μια εγγραφή μόνο όταν η τιμή *C030* εμφανίζεται στο Address Bus.

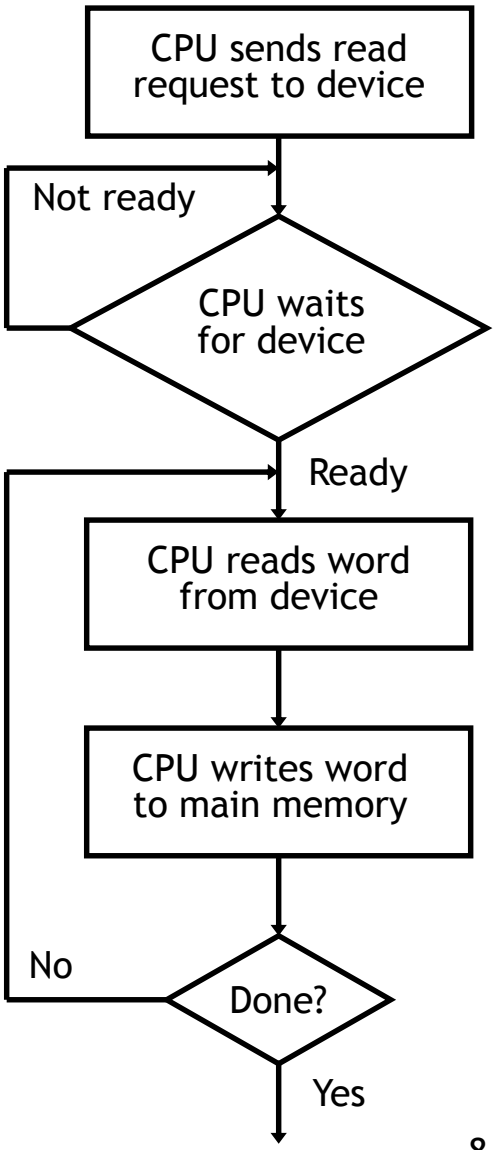
Isolated I/O

- Σε αντιδιαστολή με την προηγούμενη μέθοδο, το isolated I/O, υποστηρίζει διαφορετικούς χώρους διευθύνσεων για την κύρια μνήμη και τις περιφερειακές συσκευές. Ειδικές εντολές assembly χρησιμοποιούνται για την προσπέλαση αυτή.
- Η αρχιτεκτονική 8086 έχει 32-bit χώρο διευθύνσεων:
  - Η εντολή *MOV* χρησιμοποιείται για την προσπέλαση της κύριας μνήμης RAM.
  - Οι ειδικές εντολές IN και OUT για την προσπέλαση ενός ξεχωριστού 64KB χώρου περιφερειακών συσκευών.
  - Η διεύθυνση *0000FFFF* μπορεί να αντιστοιχεί είτε στην κύρια μνήμη είτε σε κάποια περιφερειακή συσκευή



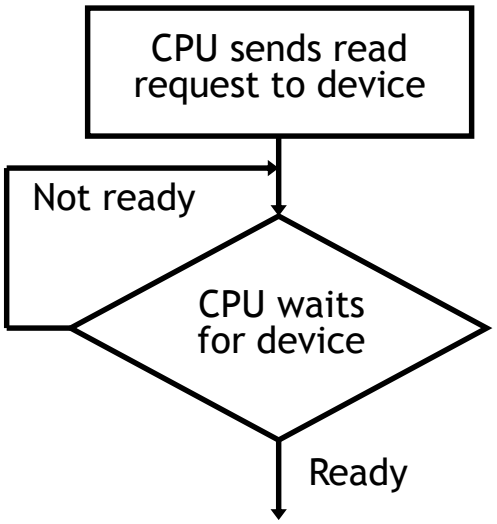
Μεταφορά Δεδομένων από Περιφερειακές Συσκευές με Polling

- Ο επεξεργαστής κάνει αίτηση για δεδομένα και περιμένει μέχρι η περιφερειακή συσκευή να είναι έτοιμη να διαβάσει ή να γράψει τα δεδομένα
  - Για μεταφορά μεγάλου όγκου δεδομένων, αυτή η διαδικασία πρέπει να επαναληφθεί πολλές φορές
- Αυτό δεν είναι και πολύ έξυπνη προσέγγιση
  - Εάν η περιφερειακή συσκευή είναι αργή σε σχέση με τον επεξεργαστή, ο επεξεργαστής περιμένει πολύ ώρα.
  - Ο επεξεργαστής μπλοκάρει περιμένοντας μην μπορώντας να κάνει τίποτε άλλο



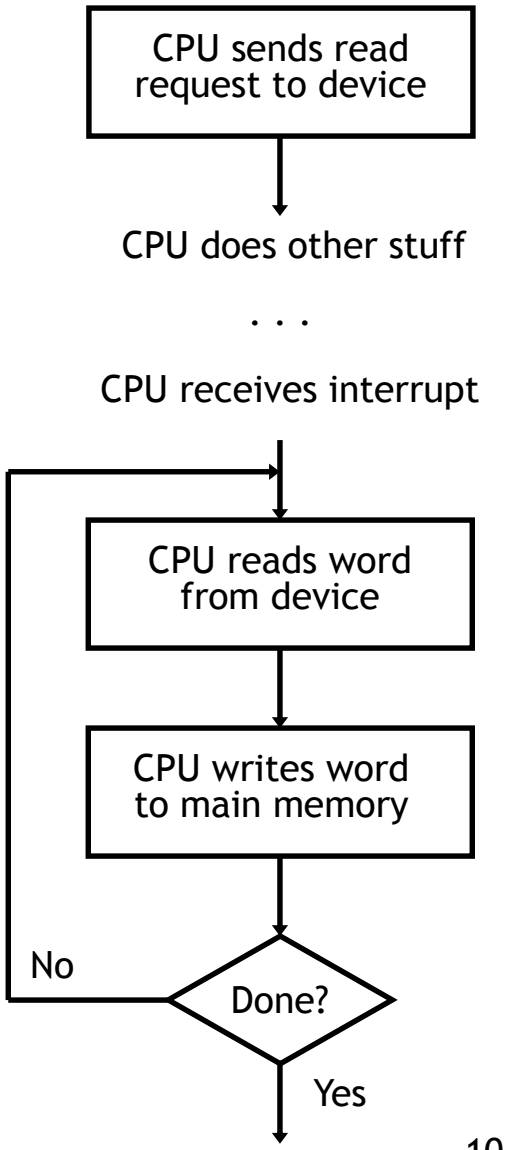
Polling

- Ο συνεχής έλεγχος για να ελέγξουμε εάν κάτι έχει λάβει χώρα ονομάζεται polling
- Δεν είναι ο καλύτερος τρόπος για να χρησιμοποιήσετε την CPU.
  - Η CPU μπλοκάρει σε ένα *while (1) loop* ελέγχοντας συνεχώς εάν η περιφερειακή συσκευή είναι έτοιμη να διαβάσει ή να στείλει δεδομένα.
- Σκεφτείτε να θέλατε να ελέγξετε εάν έχουν βγει ιατρικά αποτελέσματα από εξετάσεις που έχετε κάνει
  - Με το polling θα παίρνατε τηλ. τον γιατρό κάθε 1-2 λεπτά.
  - Μια καλύτερη ιδέα ήταν να σας τηλεφωνούσε ο γιατρός όταν τα αποτελέσματα ήταν έτοιμα



Interrupt-driven I/O

- Χρησιμοποιώντας διακοπές (interrupts) λύνουμε το πρόβλημα τις μη-αποδοτικής χρήσης του επεξεργαστή
- Μετά την αίτηση σε μια συσκευή, ο επεξεργαστής μπορεί να συνεχίσει ελεύθερα με άλλες εργασίες. Η συσκευή διακόπτει τον επεξεργαστή όταν τα δεδομένα είναι έτοιμα.
- Η μεταφορά δεδομένων μπορεί να γίνει πάλι από τον επεξεργαστή
  - *memcpy()*
- Μπορεί όμως να γίνει και με την χρήση DMA (Direct Memory Access)
  - Υλικό αυτόματης μεταφοράς δεδομένων μεταξύ I/O και μνήμης



Διακοπές - Interrupts

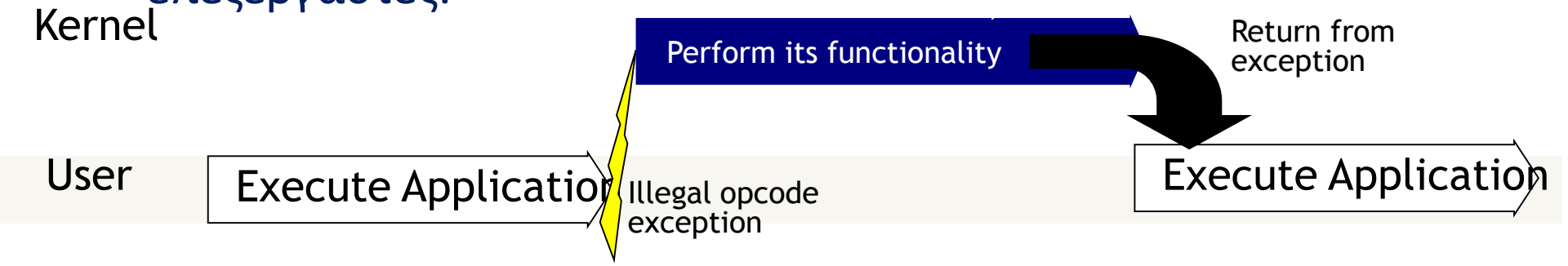
- Οι *διακοπές (Interrupts)* είναι γεγονότα που συμβαίνουν εκτός του επεξεργαστή και χρειάζονται την προσοχή του.
  - Περιφερειακές συσκευές που χρειάζονται να επικοινωνήσουν με τον επεξεργαστή δημιουργούν interrupts
  - Timers (χρονομέτρες) είναι συσκευές που ειδοποιούν ότι έχει περάσει ένα συγκεκριμένο χρονικό διάστημα.
- Οι διακοπές είναι φυσιολογικές λειτουργίες του συστήματος
  - Δεν είναι λάθη
  - Όλες οι διακοπές είναι εξυπηρετήσιμες και η κανονική ροή του προγράμματος συνεχίζει μετά την εξυπηρέτηση της διακοπής
- Το λειτουργικό σύστημα είναι υπεύθυνο για την εξυπηρέτηση της διακοπής

Εξαιρέσεις (Exceptions)

- Οι εξαιρέσεις (exceptions) είναι συνήθως λάθη κατά την διάρκεια εκτέλεσης ενός προγράμματος από τον επεξεργαστή
  - Ο επεξεργαστής προσπαθεί να εκτελέσει μια εντολή με μη-έγκυρο opcode.
  - Overflow σε μια αριθμητική πράξη ή διαίρεση με το 0.
  - Μια εντολή load ή store δεν μπορεί να ολοκληρωθεί γιατί η διεύθυνση που προσπαθεί να προσπελάσει δεν είναι στην μνήμη αλλά στον σκληρό δίσκο –Εικονική μνήμη (virtual memory) – Θα επανέλθουμε σε αυτό αργότερα.
- Υπάρχουν δύο πιθανοί τρόποι αντιμετώπισης των εξαιρέσεων
  - Εάν το λάθος ΔΕΝ είναι επανορθώσιμο, το λειτουργικό σύστημα τερματίζει το πρόγραμμα.
  - Εάν το λάθος είναι επανορθώσιμο το λειτουργικό σύστημα ή το ίδιο το πρόγραμμα μπορεί να διορθώσει την κατάσταση.

Παράδειγμα εξαίρεσης: Προσομοίωση εντολών (Instruction emulation)

- Πολλές φορές σύνολα εντολών επεκτείνονται με νέες εντολές
  - Πχ. MMX, SSE, SSE2, για τις εντολές 80x86 της Intel
- Προγράμματα που χρησιμοποιούν αυτές τις νέες εντολές δεν μπορούν να εκτελεσθούν σε παλαιότερους επεξεργαστές
  - Αυτό είναι μεγάλο πρόβλημα. Πρόβλημα “forward compatibility”.
- Ειδικό software προσομοίωσης εντολών χρησιμοποιείται για την εκτέλεση τέτοιων εντολών σε παλαιότερους επεξεργαστές.



Διακοπές και εξαιρέσεις στον MIPS

- 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

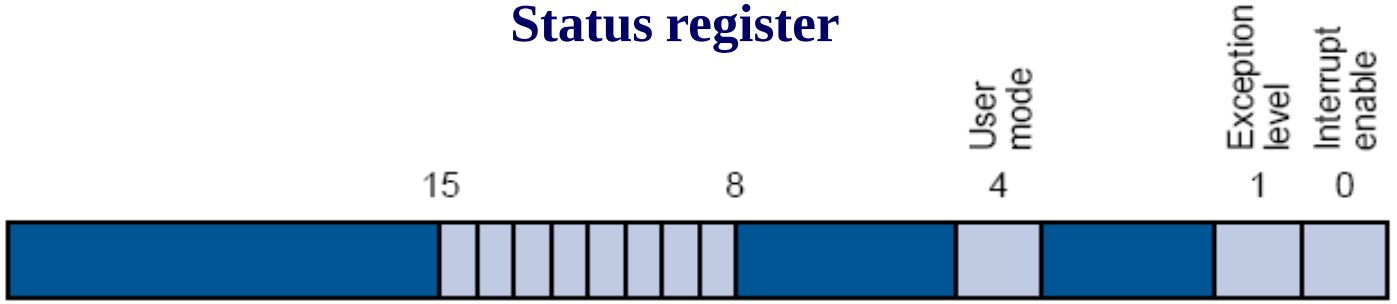
mtc0 $s0,$epc ⇔ $epc = $s0

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

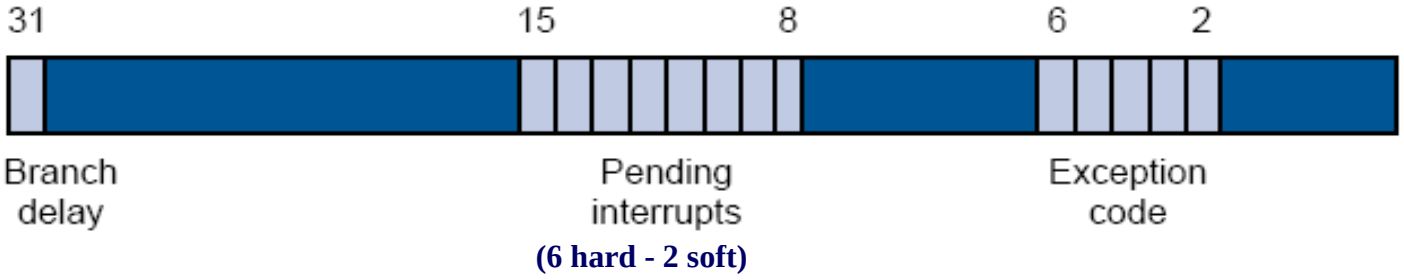
swc0 $epc,100($s1) ⇔ Mem[$s1+100] = $epc

Διακοπές και εξαιρέσεις στον MIPS

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
  - Εκκένωση του pipeline (flush) από την εντολή που δημιούργησε την εξαίρεση και όλες τις επόμενες εντολές.
  - 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

Εξαιρέσεις και Αρχιτεκτονική Διοχέτευσης

- Η εντολή που δημιούργησε την εξαίρεση θα πρέπει να εκκενωθεί (flushed) και μαζί με αυτήν όλες οι μεταγενέστερες εντολές που έχουν μπει στο pipeline.
- Η εκτέλεση θα πρέπει να συνεχιστεί με την ρουτίνα εξυπηρέτησης εξαίρεσης (handler routine) από μια προκαθορισμένη διεύθυνση μνήμης
- Εάν η ρουτίνα εξυπηρέτησης καθορίσει ότι το κανονικό πρόγραμμα μπορεί να συνεχίσει εκεί που διακόπηκε, τότε
  - Κάνει τις απαραίτητες διορθώσεις και επιστρέφει
  - Αλλιώς τερματίζει το πρόγραμμα και κάνει report το λάθος και τα αίτιά του
- Τι αλλαγές απαιτούνται από την μικρο-αρχιτεκτονική του επεξεργαστή μας για να εξυπηρετήσουμε εξαιρέσεις;

Εξαιρέσεις και μικρο-αρχιτεκτονική

- Η αντιμετώπιση των εξαιρέσεων είναι όμοια με την αντιμετώπιση control και data hazards που έχουμε ήδη περιγράψει.
- Ας θεωρήσουμε ένα παράδειγμα πρόσθεσης με υπερχείλιση που δημιουργεί εξαίρεση στο στάδιο EX του pipeline.

  `add $1, $2, $1`
- Τι πρέπει να γίνει σε αυτήν την περίπτωση;
  - Η εντολή **`add`** πρέπει να γίνει **`nop`** για να μην μεταφερθεί η λανθασμένη τιμή του **`$1`** στους καταχωρητές και στις υπόλοιπες εντολές.
  - Όλες οι εντολές πριν την **`add`** θα πρέπει να τερματιστούν κανονικά
  - Όλες οι εντολές μετά την **`add`** και η ίδια η **`add`** θα πρέπει να γίνουν flush από το pipeline (δηλ. **nops**)
  - Ή μικρο-αρχιτεκτονική θα πρέπει να θέσει τους καταχωρητές Cause και EPC με τις κατάλληλες τιμές
  - Η ρουτίνα εξυπηρέτησης εξαίρεσης (handler) θα πρέπει να αρχίζει να εκτελείται

Παράδειγμα εξαίρεσης

- Η εντολή **add** δημιουργεί εξαίρεση υπερχείλισης

```
40      sub   $11, $2, $4
44      and   $12, $2, $5
48      or    $13, $2, $6
4C      add   $1,  $2, $1
50      slt   $15, $6, $7
54      lw    $16, 50($7)
…
```

- Handler routine

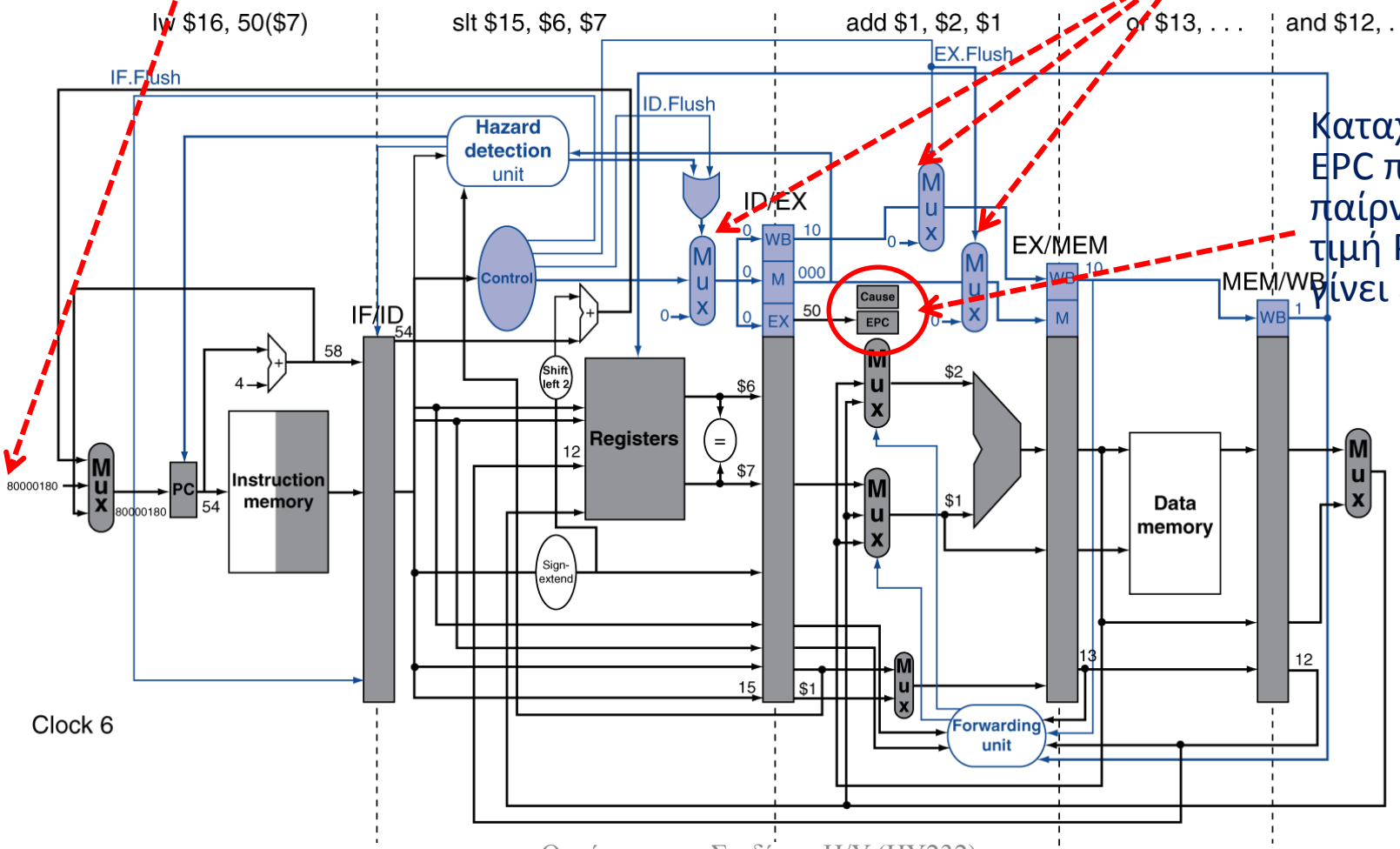
```
80000180     sw    $25, 1000($0)
80000184     sw    $26, 1004($0)
…
```

Παράδειγμα εξαίρεσης (I)

80000180 η standard διεύθυνση του handler

Extra πολυπλέκτες για να θέτουμε 0 τα σήματα ελέγχου (flush) στο στάδιο EX και στο στάδιο ID.

Καταχωρητής EPC που παίρνει την τιμή PC+4 όταν γίνει εξαίρεση



Παράδειγμα εξαίρεσης (II)