Ψηφιακοί Υπολογιστές

΄Υλη για Input/Output (I/O), polling, interrupts

Σκοπός

- Σκοπός του εργαστηρίου
  - Διαχείριση συσκευών I/O (πληκτρολόγιο, οθόνη/κονσόλα) με χρήση Memory Mapped I/O
  - Κατανόηση της λειτουργίας των περιφερειακών με χρήση polling και interrupts
- Μέρη εργαστηρίου
  - Memory Mapped I/O και τεχνική polling
  - Memory Mapped I/O και χρήση interrupts

Memory Mapped I/O (1/2)

- Επικοινωνία επεξεργαστή – περιφερειακών
  - Χρήση απλών εντολών ανάγνωσης/εγγραφής συγκεκριμένων θέσεων μνήμης
- 2 μονάδες I/O
  - Πληκτρολόγιο
  - Κονσόλα
- 2 διευθύνσεις
  - Control
  - Data

Memory Mapped I/O (2/2)

- Data
  - 8 least significant bits
  - ASCII character
- Control
  - 2 least significant bits
  - Bit 0: Ready bit
    - Αν η συσκευή είναι έτοιμη για λειτουργία
  - Bit 1: Interrupt enable
    - Ενεργοποιεί τη λειτουργία των interrupts

Διευθύνσεις Μνήμης Περιφερειακών

Όνομα Καταχωρητή	**Διεύθυνση**
Receiver Control	0xffff0000
Receiver Data	0xffff0004
Transmitter Control	0xffff0008
Transmitter Data	0xffff000c

- Receiver
  - Πληκτρολόγιο
- Transmitter
  - Κονσόλα

Τεχνική Polling

- Εμφάνιση ενός χαρακτήρα στην κονσόλα
- Διαδικασία
  - Ανάγνωση Transmitter Control (0xffff0008)
  - Αν Ready bit == 0 πάμε ξανά στο παραπάνω step
  - Εγγραφή στον Transmitter Data (0xffff000c) το χαρακτήρα (1 Byte) που θέλουμε να εμφανίσουμε
- Ανάγωση ενός χαρακτήρα (??)

Ζητούμενο 1ο Μέρος (1/2)

- Δύο συναρτήσεις
  - write_ch
  - read_ch
- Χρήση τεχνικής polling
  - εμφάνιση ενός χαρακτήρα στην κονσόλα
  - ανάγνωση ενός χαρακτήρα από το πληκτρολόγιο
- **ΠΡΟΣΟΧΗ!!!**
  - **ΟΧΙ SYSCALL!!!**

Ζητούμενο 1ο Μέρος (2/2)

- Δύο συναρτήσεις
  - write_string
  - read_string
- Χρήση των δύο συναρτήσεων, read_ch και write_ch, επαναληπτικά για ολόκληρα strings
- Δομή κώδικα:

Loop:

εκτύπωση μηνύματος εισαγωγής string

read_str

write_str

εναλλαγή πεζών-κεφαλαίων

write_str

j loop

Interrupts

- Διαφορετικός τρόπος επικοινωνίας επεξεργαστή-περιφερειακών συσκευών
- Λειτουργία
  - Κανονική λειτουργία επεξεργαστή
  - **Interrupt!!! Εκτέλεση κώδικα Interrupt handler**
  - Συνέχεια λειτουργίας κώδικα

Interrupt handler

- Κώδικας: exceptions.s
- Χρήση όλων των καταχωρητών αλλά κυρίως
  - $k0 και $k1
- # Interrupt-specific code goes here!
- Μετονομάστε το αρχείο exceptions.s σε exceptions_lab6_b.s και ορίστε το στο εργαλείο SPIM

Ενεργοποίηση interrupts

- Coprocessor0, καταχωρητής 12 ($12)
  - Bit 11 ενεργοποιεί τις διακοπές από το πληκτρολόγιο
  - Bit 0 ενεργοποιεί τις διακοπές για τον επεξεργαστή

```
mfc0 $t0, $12
li $t1, 0x801
and $t0, $t0, $t1
addi $t0, $t0, 1
mtc0 $t0, $12
```

- Receiver control
  - Bit 1: Interrupt enable

```
li $t0, 0xffff0000
lw $t1, 0($t0)
ori $t2, $t1, 0x2
sw $t2, 0($t0)
```

Ζητούμενο 2ο Μέρος (1/2)

- Δημιουργία 2 θέσεων μνήμης cdata και cflag
- Αρχικοποίηση θέσης μνήμης cflag με την τιμή 0
- Ενεργοποίηση Interrupts
- Εμφάνιση μηνύματος εισαγωγής χαρακτήρα
- Loop:
  - Έλεγχος αν το cflag == 1
    - Αν ναι, διάβασε το cdata. Διαφορετικά συνέχισε την ανάγνωση του cflag και κάνε τον έλεγχο ξανά.
    - Αν το cdata είναι κενό, τερματισμός προγράμματος
    - Διαφορετικά εκτύπωσε στην οθόνη τον χαρακτήρα
    - Πήγαινε ξανά στην εμφάνιση μηνύματος εισαγωγής χαρακτήρα

Ζητούμενο 2ο Μέρος (2/2)

- Κώδικας Interrupt handler:
  - # Interrupt-specific code goes here!
  - Ανάγνωση εισερχόμενου χαρακτήρα από το receiver data
  - Αλλαγή του χαρακτήρα από πεζό-κεφαλαίο ή το αντίστροφο
  - Άποθήκευση νέου χαρακτήρα στη μνήμη (cdata)
  - Αποθήκευση της τιμής 1 στη μνήμη cflag