20/12/2015

Αναφορά 5ου εργαστηρίου

Προεργασία

Σ' αυτό το εργαστήριο κληθήκαμε να δημιουργήσουμε 2 συναρτήσεις ,οι οποίες θα γράφουν και θα διαβάζουν έναν χαρακτήρα από το πληκτρολόγιο , με βάση την τεχνική polling.Εν συνεχεία χρησιμοποιώντας αυτές τις συναρτήσεις ,και όχι syscall, θα δημιουργήσουμε ένα πρόγραμμα μετατροπής μιας συμβολοσειράς σε κεφάλαια. Το πρόγραμμα αυτό, με βάση το πλάνο μας , θα περιέχει τρεις συναρτήσεις εκ των οποίων η μια θα διαβάζει μια συμβολοσειρά από το πληκτρολόγιο ή μια έτοιμη και ορισμένη από εμάς, η άλλη θα τη μετατρέπει σε κεφάλαια , και η τελευταία θα την εμφανίζει διαμορφωμένη στην οθόνη. Για να δημιουργήσουμε τον κώδικα αυτό , χρειάστηκε να κατανοήσουμε πως συνδέονται οι περιφερειακές συσκευές με τον επεξεργαστή. Πιο συγκεκριμένα , μάθαμε για τις διευθύνσεις μνήμης Data και Control που χρειάζονται για τη μεταφορά δεδομένων και τον έλεγχο της συσκευής αντίστοιχα. Τέλος , μάθαμε για την τεχνική polling καθώς και τι συμβαίνει στις διευθύνσεις μνήμης transmitter/receiver data και transmitter/receiver control κατά την ανάγνωση και μεταφορά δεδομένων.

Περιγραφή ζητούμενων

Ο σκοπός του εργαστηρίου είναι να κατανοήσει η ομάδα μας τον τρόπο με τον όποιο γράφονται και διαβάζονται χαρακτήρες από το πληκτρολόγιο προς την κονσόλα χωρίς syscall, καθώς και να εμβαθύνει σε όλα αυτά. Υλοποιώντας αυτό το εργαστήριο μάθαμε ότι τα 8 λιγότερο σημαντικά bits των καταχωρητών Data και τα δυο λιγότερο σημαντικά bits των καταχωρητών Control , είναι αυτά που χρησιμοποιούνται για την εκτέλεση των λειτουργιών του προγράμματος (μεταφορά δεδομένων και έλεγχου περιφερειακών συσκευών). Τέλος , σημαντικό είναι να καταλάβουμε ότι η τεχνική polling είναι μια σειρά διαρκών ενεργειών για τον έλεγχο των συσκευών ,μέχρι να καταλάβουμε ότι αυτές οι συσκευές είναι έτοιμες για να δεχθούν ή να πάρουν δεδομένα.

Περιγραφή εκτέλεσης

Το πρώτο βήμα που κάναμε για να υλοποιήσουμε το πρόγραμμα ήταν να δημιουργήσουμε τις συναρτήσεις Write_ch και Read_ch με βάση την τεχνική

polling, καθώς και τη συνάρτηση main που τις καλεί, με σκοπό να ελέγξουμε την ορθή λειτουργία τους. Οι συναρτήσεις που φτιάξαμε είναι οι εξής :

Read_ch	Write_ch

Read_ch:
lb $t0,0xffff0000
andi $t0,$t0,1
beq $t0,$0,Read_ch
lb $v0,0xffff0004
jr $ra
	Write_ch:
	lw $t0, 0xffff0008
	andi $t0, $t0, 1
	beq $t0,$0,Write_ch
	sw $a0, 0xffff000c
	jr $ra

Η συνάρτηση αποθηκεύει στον καταχωρητή $t0 την διεύθυνση του Receiver Control με σκοπό να ελέγξει την τιμή του λιγότερο σημαντικού bit (ready bit). Στη συνέχεια με τη λογική πράξη and κρατά στον ίδιο καταχωρητή το ready bit. Αν το ready bit είναι 0, γίνεται μια επανάληψη έως ότου γίνει 1. Αν είναι ή γίνει 1 η τιμή του, τότε ο χαρακτήρας μπαίνει στην διεύθυνση 0xffff0004.

Έπειτα ο χαρακτήρας αυτός αφού αποθηκεύτηκε στον πίνακα String[80] το ίδιο θα γινόταν και με τους υπόλοιπους.Σειρά μετά είχε η συνάρτηση UpperConv η οποία παίρνει ως όρισμα τον πίνακα String[80] και αφού προσπελάσει τον πίνακα για να βρει το μέγεθος που θέλαμε (μια επανάληψη με ένα μετρητή μέχρι τον χαρακτήρα \0) κάναμε τον παρακάτω έλεγχο (θα δειχθεί σε C, σε Assembly MIPS θα υπάρχει στο παράρτημα):

```
int i,len; // len is the counter of last loop
char string[80];

while(i<len){
        if(string[i] < 97 && string[i] > 122)  string[i] = string[i] - 32;
        i++;
}
```

Συμπέρασμα

Τέλος, μέσω αυτού του εργαστηρίου κατανοήσαμε πως επιτυγχάνεται η επικοινωνία του επεξεργαστή με δυο περιφεριακές συσκευές το πληκτρολόγιο σαν είσοδο και την κονσόλα του PCSpim σαν έξοδο.Αν και δεν δουλέψαμε τον δεύτερο μέρος της άσκησης δηλαδή η ίδια δουλειά αλλά με άλλο τρόπο.Φυσικά δουλεύοντας με interrupts και exceptions δεν είναι πιο εύκολη αλλά πιο ανώδυνη για τον επεξεργαστή μας.Δηλαδή αυτό που καταλάβαμε ήταν ότι με την τεχνική των interrupts ο επεξεργαστής όχι μόνο γλιτώνει κάποιες περιττές επαναλήψεις αλλά και παύει να δουλεύει αυτός και δουλεύει ο interrupt handler ο οποίος επικοινωνεί

αυτός με αυτές τις συσκευές.Εν τέλει το συμπέρασμα είναι ότι η τεχνική polling δεν είναι και πολύ πρακτική.

Παράρτημα – Κώδικες

```
.data
.globl main
.globl Print_String
.globl Read_String
.globl Write_ch
.globl Read_ch
.globl UpperConv
.globl string

mess:        .asciiz "Please give the string: "
mes:     .asciiz "Converted to: "
newline:    .asciiz "\n"

string:  .align 2
                .space 80

.text

main:

          la $a0, mess
          jal Print_String
```

```
        la $a0, string
        jal Read_String

        la $a0, string

        la $a0,newline
        jal Print_String

        la $a0, mes
        jal Print_String

        la $a0, string

        jal UpperConv

        move $a0,$v0
        jal Print_String

        li  $v0, 10
        syscall

################################################################

Print_String:

        addi $sp, $sp, 4
        sw $ra, 0($sp)
```

```
        move $t2, $a0
        li $t1, 0
loop_print:
        add $t3, $t1, $t2
        lb $a0, 0($t3)
        jal Write_ch

        beq $a0, 0, after_loop_print
        beq $a0, 10, after_loop_print

        addi $t1, $t1, 1
        j loop_print
after_loop_print:

        lw $ra, 0($sp)
        addi $sp, $sp, -4
        jr $ra

###############################################################
Read_String:

        addi $sp, $sp, 4
        sw $ra, 0($sp)
        la $s0, string
        li $t5, 0

read:
```

```
        jal Read_ch
        sb $v0, string($t5)

        beq $v0,0,after_read_loop
        beq $v0,10,after_read_loop

        addi $t5,$t5, 1

        move $a0, $v0

        jal Write_ch

        j read

after_read_loop:


        lw $ra, 0($sp)
        addi $sp, $sp, -4
        jr $ra


#################################################################
Write_ch:

        lw $t0, 0xffff0008

        andi $t0, $t0, 1
```

```
        beq $t0,$0,Write_ch
        sw $a0, 0xffff000c

        jr $ra

################################################################

Read_ch:

        lb $t0,0xffff0000

            andi $t0,$t0,1
        beq  $t0,$0,Read_ch
        lb   $v0,0xffff0004

        jr $ra

################################################################

UpperConv:

        addi $sp $sp, 4
        sw $ra, 0($sp)
        li $t2, 0
        li $t3, 0
        li $t4, 0

Upper_loop:
```

```
        bge $t3, 80 , end_upper_loop
        add $t5, $a0, $t3
        lb $t6, 0($t5)
        beq $t6, 3, end_if_upper
        addi $t4, $t4, 1
end_if_upper:

        addi $t3, $t3, 1
        j Upper_loop

end_upper_loop:

        li $t5, 0
while_loop_upper:
        bge $t2, $t4, end_while_loop_upper
        add $t5, $a0, $t2
        lb $t7, 0($t5)
        blt $t7, 97, end_if_upper_3
        bgt $t7, 122, end_if_upper_3
        sb $t7, 0($t5)
end_if_upper_3:

        addi $t2, $t2, 1
        j while_loop_upper
end_while_loop_upper:

        la $v0,string
```

```
        lw $ra, 0($sp)
        addi $sp, $sp, -4
        jr $ra

#################################################################
```