ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

Τομέας Υλικού και Αρχιτεκτονικής Υπολογιστών

ΗΥ134 - Εισαγωγή τους Η/Υ

Εργαστηριακή Άσκηση 2

Εαρινό εξάμηνο 2010

Ο σκοπός της 2ης εργαστηριακής άσκησης είναι να εξοικειωθείτε με τις εντολές εισόδου-εξόδου για χρήση της κονσόλας καθώς και με την χρήση συνθηκών σε δομές επιλογής (τα γνωστά if) και σε δομές επανάληψης (for loop). Σε αυτό το αρχείο θα βρείτε μια επεξήγηση των Assembly εντολών που υλοποιούν τις δομές ελέγχου ροής και τις αντίστοιχες I/O λειτουργίες.

- **Εντολές για είσοδο ακεραίων αριθμών από την κονσόλα:**

```
addi $v0, $0, 5
syscall
add $t0, $v0, $0  # καταχώριση της τιμής που εισάγαμε, στον register $t0
```

- **Εντολές για εμφάνιση ακεραίου στην κονσόλα:**

```
li $v0, 1    # pseudo-instruction για addi $v0, $0, 1
add $a0, $t0, $0 # ο $t0 έχει την τιμή που θέλουμε να εμφανίσουμε
syscall
```

- **Εντολές για εμφάνιση μηνυμάτων στην κονσόλα:**

Για να τυπώσετε ένα string στην οθόνη θα πρέπει να το δηλώσετε πρώτα στον τμήμα .data του προγράμματός σας ως εξής:

Αρχικά πρέπει να ορίσετε το string στην περιοχή της μνήμης με ένα label (στην περίπτωση του παραδείγματος String_name1) για να μπορείτε να έχετε πρόσβαση με χρήση της διεύθυνσης του.

```
. data
String_name1:.asciiz  “μήνυμα”
```

Σε περίπτωση που θέλετε να αλλάξετε γραμμή (για να είναι πιο ευανάγνωστη η εκτέλεση) πρέπει να ορίσετε ένα string αλλαγής γραμμής.

```
String_name2:.asciiz  “\n”
```

Για να τυπώσετε το string στην κονσόλα πρέπει να γράψετε τις εξής εντολές.

```
addi $v0, $0, 4
la   $a0, String_name1
syscall
```

Οι εντολές syscall είναι κλήσεις λειτουργιών του συστήματος. Με αυτές μπορείτε εκτός από I/O λειτουργίες να τερματίσετε ένα πρόγραμμα (syscall με παράμετρο $v0=10) ή να ζητήσετε δυναμικά μνήμη (αντίστοιχη εντολή malloc της C). Μπορείτε να βρείτε όλες τις λειτουργίες στο instruction set αρχείο που υπάρχει στο site του μαθήματος.

Υλοποίηση δομών έλεγχου ροής προγράμματος.

Εντολές έλεγχου ροής είναι το σύνολο των εντολών το οποίο αλλάζουν την ροή εκτέλεσης προγράμματος. Κανονικά η σειρά εκτέλεσης των εντολών είναι μετά από κάθε εντολή να εκτελείται η αμέσως από κάτω. Υπάρχουν όμως εντολές οι οποίες αλλάζουν είτε στατικά (χωρίς συνθήκη) είτε δυναμικά (με συνθήκη) την ροή εκτέλεσης των εντολών.

Παράδειγμα ελέγχου ροής, μετατροπή από C σε assembly:

Παράδειγμα 1 (if)

```
if( x = = 0){
.
.
                //Case true
.

}
else
{
.
.
                //Case false
.
}
```

Αντιστοιχίζουμε την μεταβλητή x στον καταχωρητή $t3

```
bnez $t3,else
.
.
                        #case true
.
j endif
else:
.
.
                        #case false
```

.
endif:

Παράδειγμα 2 (for)

```
for( i =0 ; i < 10 ; i++ )
{
        ….              //code
}
```

Αντιστοιχίζουμε την μεταβλητή i στον καταχωρητή $t3

```
add $t3,$0,$0
addi $t4,$0,10

for: bge $t3,$t4,endfor

…               #code

addi $t3,$t3,1
j for
endfor:
```

Παράδειγμα 3 (multiple if)

```
if(x>0 && x<10)
{
        …       //code
}
```

Αντιστοιχίζουμε την μεταβλητή x στον καταχωρητή $t3

```
addi $t4,$0,10
blez $t3,endif
bge $t3,$t4,endif

        …       #code

endif:
```

β' τρόπος

```
addi $t4,$0,10
bgtz $t3,cond2# if(x>0)
j endif
cond2:
blt $t3,$t4,is_true       # if(x<10)
j endif
is_true:
```

```
…       #code
```

endif:

Το πλήθος των εντολών είναι σχετικά μεγάλο αλλά η λειτουργία τους είναι πολύ απλή. Συνιστάται κατά τον προγραμματισμό σε assembly να κάνετε χρήση του instructions set. Μπορείτε επίσης να χρησιμοποιήσετε και ψευδο-εντολές σαν να ήταν κανονικές εντολές (γιά παράδειγμα li, la, bgez, blt κοκ). Η αποστήθιση όλων των εντολών δεν είναι ζητούμενο σε αυτό το εργαστήριο ωστόσο πρέπει να είσθε σε θέση να μπορείτε να χρησιμοποιήσετε όλες τις εντολές που αναγράφονται σε αυτό το αρχείο.

Για το επόμενο εργαστήριο έχετε να υλοποιήσετε 3 εργαστηριακές ασκήσεις. Την ώρα του εργαστηρίου θα εξετασθείτε προφορικά πάνω στους κώδικες που θα παραδώσετε.

Εκφώνηση 1ης εργαστηριακής άσκησης:

Μετατρέψτε σε assembly τον παρακάτω κώδικα C.

```
if( x = = 0 || y = = 0 )
        printf("product zero\n");
else if( x > 0 && y > 0 )
        printf("product positive\n");
else if( x < 0 && y < 0 )
        printf("product positive\n");
else
        printf("product negative\n");
```

όπου x, y 2 προσημασμένοι ακέραιοι που θα δίνει ο χρήστης σαν είσοδο.
Παραδείγματα της κονσόλας μετά από την εκτέλεση ενός τέτοιου προγράμματος:

```
Please give an integer:8
Please give an integer:-3
product negative


Please give an integer:-2
Please give an integer:-4
product positive


Please give an integer:5
Please give an integer:0
product zero
```

Εκφώνηση 2ης εργαστηριακής άσκησης:

Υλοποιήσατε ένα πρόγραμμα σε assembly το οποίο αρχικά θα ζητά από τον χρήστη να εισάγει έναν θετικό ακέραιο αριθμό (n>0). Ο χρήστης θα εισάγει από την κονσόλα τον αριθμό και το πρόγραμμα θα ελέγχει αν ο αριθμός καλύπτει την συνθήκη αυτή. Αν όχι και για όσο δεν καλύπτεται η συνθήκη θα ζητά από τον χρήστη να δώσει τον αριθμό. Όταν δοθεί θετικός αριθμός που καλύπτει την συνθήκη τότε το πρόγραμμα θα υπολογίζει και θα τυπώνει το άθροισμα $\sum_1^n i$ και θα τερματίζει. Ο υπολογισμός θα πρέπει να γίνεται με την χρήση for loop.

Ένα παράδειγμα της κονσόλας μετά από την εκτέλεση ενός τέτοιου προγράμματος είναι το εξής:

Please give an positive integer >0: -2 εκτός αποδεκτών ορίων
Please give an positive integer >0: 0 εκτός αποδεκτών ορίων
Please give an integer >0: 5

sum=15

Εκφώνηση 3ης εργαστηριακής άσκησης:

Να υλοποιήσετε πρόγραμμα σε assembly το οποίο θα διαβάζει έναν αριθμό (θεωρήστε ότι θα εισάγετε θετικό αριθμό) και θα υπολογίζει και θα εμφανίζει το πλήθος των περιττών μέχρι τον αριθμό αυτό. Αν για παράδειγμα εισάγουμε τον αριθμό 11 τότε οι άρτιοι μέχρι το 11 είναι (1,3,5,7,9,11) άρα πρέπει να εμφανίσει πλήθος=6.

Φροντίστε οι κώδικες που θα παρουσιάσετε στην εξέταση να είναι ευανάγνωστοι (διαχωρισμός των επιμέρους κομματιών) και επαρκώς σχολιασμένοι.