Εργαστήριο 2

Εαρινό Εξάμηνο 2010-2011

Στόχοι του εργαστηρίου

- Εντολές εισόδου-εξόδου
- Χρήση συνθηκών σε δομές επιλογής
- Χρήση συνθηκών σε δομές επανάληψης

Εντολές εισόδου-εξόδου

Εντολές για είσοδο ακεραίων αριθμών από την κονσόλα:

```
addi $v0, $0, 5        # κωδικός 5 από τα syscall στον $v0
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
                       # κωδικός 1 από τα syscall στον $v0
add $a0, $t0, $0       # ο $t0 έχει την τιμή που θέλουμε να
                       # εμφανίσουμε και το φορτώνουμε στον $a0
                       # σαν παράμετρο
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              # κωδικός 4 από τα syscall στον $v0
la   $a0, String_name1       # φορτώνουμε στον $a0 την διεύθυνση
                             # του String που θα εκτυπώσουμε
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 user guide 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.
}
…           // Next instruction
```

Αντιστοιχίζουμε την μεταβλητή x στον καταχωρητή $t3

```
bnez $t3,else
…                # case true
j endif

else:
…                # case false

endif:
…                # next instruction
```

Παράδειγμα 2 - for

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

Αντιστοιχίζουμε την μεταβλητή i στον καταχωρητή $t3 και αποθηκεύουμε το όριο του loop (εδώ 10) στον καταχωρητή $t4.

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

for:
bge $t3,$t4,endfor

…           # code

addi $t3,$t3,1
j for

endfor:
…           # next instruction
```

Παράδειγμα 3 - multiple if

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

Αντιστοιχίζουμε την μεταβλητή x στον καταχωρητή $t3 και αποθηκεύουμε την σταθερά με την οποία θα συγκρίνουμε (εδώ 10) στον καταχωρητή $t4.

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

…     # code

endif :
…     # next instruction
```

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:
…     # next instruction
```

Το πλήθος των εντολών είναι σχετικά μεγάλο αλλά η λειτουργία τους είναι πολύ απλή. Συνιστάται κατά τον προγραμματισμό σε assembly να κάνετε χρήση του instruction set. Μπορείτε επίσης να χρησιμοποιήσετε και ψευδο-εντολές σαν να ήταν κανονικές εντολές (για 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 τις εντολές που αναγράφονται σε αυτό το αρχείο.

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

Άσκηση 1 (3 μονάδες)

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 θετικών πολλαπλάσιων του 3 μέχρι τον αριθμό αυτό. Αν για παράδειγμα εισάγουμε τον αριθμό 16 τότε οι πολλαπλάσιοι του 3 μέχρι το 16 είναι (3,6,9,12,15) άρα πρέπει να εμφανίσει πλήθος=5.

Άσκηση 2 (3 μονάδες)

Υλοποιήσατε ένα πρόγραμμα σε assembly το οποίο αρχικά θα ζητά από τον χρήστη να εισάγει έναν αρνητικό 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 αρνητικός αριθμός που καλύπτει την συνθήκη τότε το πρόγραμμα θα υπολογίζει και θα τυπώνει το άθροισμα $\sum_{n}^{-1} i$ 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 a negative integer < 0: 2 εκτός αποδεκτών ορίων
Please give a negative integer < 0: 0 εκτός αποδεκτών ορίων
Please give a negative integer < 0: -5

sum = -15
```

Άσκηση 3 (4 μονάδες)

Να υλοποιήσετε σε assembly ένα πρόγραμμα το οποίο θα ζητά από τον χρήστη να μαντέψει έναν ακέραιο αριθμό μεταξύ 0 και 100. Κάθε φορά, επαναληπτικά μέχρι που να βρεθεί, θα ζητά στον χρήστη να εισάγει έναν ακέραιο αριθμό και θα τον καθοδηγήσει απαντώντας ανάλογα:

- Αν ο αριθμός που έδωσε ο χρήστης είναι μικρότερο από αυτό που ψάχνουμε, τότε θα εμφανίσει μήνυμα «Too small!»
- Αν ο αριθμός που έδωσε ο χρήστης είναι μεγαλύτερο από αυτό που ψάχνουμε, τότε θα εμφανίσει μήνυμα «Too big!»
- Αν ο χρήστης βρήκε τον αριθμό, τότε εμφανίζει κατάλληλο μήνυμα και εμφανίζει επίσης το πλήθος των προσπαθειών που χρειάστηκαν μέχρι που να το βρει.

Επίσης, μετά από μήνυμα τύπου «Too small!» ή «Too big!», θα πρέπει να εμφανίσει μήνυμα που να προσδιορίζει πόσο κοντά βρίσκεται ο χρήστης από τον αριθμό που ψάχνουμε. Θεωρήστε ότι υπάρχουν 3 πιθανά μηνύματα: μακριά (διαφορά ≥ 25), κοντά (10 ≤ διαφορά < 25), πολύ κοντά (διαφορά < 10).

Παράδειγμα εκτέλεσης: (ψάχνοντας για το 78)

```
Try to guess which number between 0 and 100 i am thinking of!
Please give an integer :
**** user input : 2
Too small! But you are far away!
Please give an integer :
**** user input : 65
Too small! But you are close!
Please give an integer :
**** user input : 79
Too big! But you are very close!
Please give an integer :
**** user input : 78
You guessed it with 4 tries!
-- program is finished running --
```

Θα πρέπει να στέλνετε με email τις λύσεις των εργαστηριακών ασκήσεων σας στους διδάσκοντες στο ce134lab@gmail.com.

Το email σας θα πρέπει να περιέχει ως attachment **ένα zip file** με τον κώδικα σας.

Κάθε διαφορετική άσκηση στην εκφώνηση θα βρίσκεται και σε διαφορετικό asm file. **Το όνομα των asm files θα ΠΡΕΠΕΙ να αρχίζει με το ΑΕΜ σας.**
Για παράδειγμα, το lab2.zip θα περιέχει 3 asm files, ένα για κάθε μία από τις ασκήσεις του lab2, με ονόματα 999_lab2a.asm, 999_lab2b.asm, 999_lab2c.asm για τον φοιτητή με ΑΕΜ 999.

Το email σας θα έχει Subject: CE134, lab N, Section X (N ο αριθμός του lab, N=2 ..., και X=1 έως 7).
Το email σας θα έχει body: το όνομα σας και το ΑΕΜ σας.

Θα πρέπει να στέλνετε το email σας πριν βγείτε από την εξέταση του εργαστηρίου.