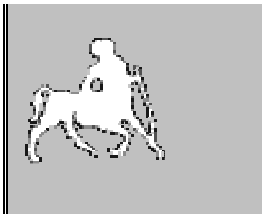

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

ΗΥ134 - Εισαγωγή στον Σχεδιασμό και Οργάνωση Η/Υ Ι

Εργαστηριακή Άσκηση 3

Εαρινό εξάμηνο 2010

Σκοπός της 3ης εργαστηριακής άσκησης είναι να εξοικειωθείτε με τις εντολές προσπέλασης της μνήμης. Θα δούμε πως ορίζουμε μια δομή δεδομένων (πίνακα) στην μνήμη και πως μπορούμε να γράψουμε ή να διαβάσουμε δεδομένα σε αυτήν.

Χρήση μνήμης.
Τα δεδομένα αποθηκεύονται σε ξεχωριστό τμήμα μνήμης το οποίο χρησιμοποιείται αποκλειστικά για αυτό το σκοπό. Γι' αυτό τον λόγο η δήλωση του πίνακα, που θα χρησιμοποιήσετε, πρέπει να γίνει στο τμήμα **.data.** Ο τρόπος δήλωσης του πίνακα και η σύνταξη των εντολών load (lw) και store (sw), οι οποίες χρησιμοποιούνται για την μεταφορά δεδομένων μεταξύ μνήμη και καταχωρητή, εξηγούνται παρακάτω.

- **Δήλωση πίνακα**

```
.data                          # στο τμήμα .data πρέπει να δηλωθεί ο πίνακας

Name_array: .space X    #   Name_array: δώστε ότι όνομα θέλετε στον πίνακα
                              #   .space X:  δηλώνει ότι θέλουμε να κρατήσουμε
                              #   χώρο ίσο με X byte για τον πίνακα Name_array
```

- **Εντολή load (lw):** Η εντολή αυτή αποθηκεύει δεδομένα σε καταχωρητή, τα οποία έχει πάρει από συγκεκριμένη διεύθυνση της μνήμης.

Σύνταξη: **lw Rt, Address(Rs)** Σημασία**: Rt=Memory[Address+Rs]**

Προσοχή: όπου Rt, Rs είναι καταχωρητές και όπου Address είναι το όνομα του πίνακα (label) που δώσατε στο τμήμα .data.
Με βάση την παραπάνω δήλωση (στο τμήμα .data) θα έπρεπε να γράψουμε:
lw Rt, Name_array(Rs)

Από την σύνταξη της εντολής αυτής γίνεται κατανοητό ότι ζητάμε τα δεδομένα της μνήμης στην διεύθυνση [Name_array+Rs] και τα οποία θα αποθηκευτούν στον καταχωρητή Rt.

- **Εντολή store (sw):** Η εντολή αυτή αποθηκεύει δεδομένα από έναν καταχωρητή σε συγκεκριμένη διεύθυνση στη μνήμη.

Σύνταξη: **sw Rt, Address(Rs)** Σημασία: **Memory[Address+Rs]=Rt**

- Ο τρόπος λειτουργίας είναι παρόμοιος με την εντολή **lw(load word).** Η λειτουργία που υλοποιεί αυτή η εντολή είναι: Αποθήκευσε τα περιεχόμενα του καταχωρητή Rs στην μνήμη και συγκεκριμένα στην διεύθυνση **[Address+Rs].**

Η διάφορα με την πρώτη εντολή είναι ότι η sw αποθηκεύει δεδομένα στην μνήμη ενώ η lw διαβάζει τα δεδομένα της μνήμης.

Ευθυγράμμιση

Όπως γνωρίζετε από την θεωρεία ο MIPS είναι ένας 32bit επεξεργαστής. Αυτό συνεπάγεται ότι όλοι οι καταχωρητές του έχουν μέγεθος 32 bits ή αλλιώς 4 bytes. Ορίζουμε σαν **word** τα δεδομένα μεγέθους 4 bytes τα οποία χωράνε σε έναν register.

Όταν ορίζουμε έναν πίνακα στην μνήμη με την εντολή **.space** καθορίζουμε το μέγεθος του ίσο με έναν αριθμό από **bytes** όπως παρουσιάστηκε πριν. Εφόσον όμως τα στοιχεία που αποθηκεύονται σε αυτόν θα είναι μεγέθους 4 bytes το καθένα, πρέπει να καθορίσετε το μέγεθος του πίνακα να είναι πολλαπλάσιο του 4. Έτσι το πρώτο στοιχείο του πίνακα βρίσκεται στα bytes 0 έως 3. Αν θέλετε να το κάνετε load θα πρέπει να συντάξετε την εντολή lw Rt,Address(Rs) με τον Rs να έχει περιεχόμενο 0. Αν τώρα θέλετε το δεύτερο στοιχείο του πίνακα αυτό βρίσκεται στα bytes 4 έως 7. Για να το κάνετε load θα πρέπει να γράψετε την ίδια εντολή αλλά ο Rs θα πρέπει να έχει περιεχόμενο τον αριθμό 4.

Σημείωση: Όταν ορίζετε έναν πίνακα πρέπει να δηλώνετε και τι μέγεθος θα έχουν τα δεδομένα που θα αποθηκεύονται σε αυτόν. Αυτό γίνεται με την εντολή **.align n** όπου **n** έναν ακέραιος που θα δώσετε εσείς. Η εντολή αυτή σημαίνει ότι τα στοιχεία του πίνακα έχουν μέγεθος 2^n. Έτσι για έναν πίνακα 10 θέσεων με λέξεις 4 bytes πρέπει να συντάξετε την εντολή:

```
.align 2          #μέγεθος στοιχείου 4 bytes
label: .space 40  #μέγεθος πίνακα 40 bytes = 10 στοιχεία
```

Ένα παράδειγμα χρήσης μνήμης.

```
.data
.align 2                #λέξεις 4 bytes
vector: .space 24       #πίνακας 6 θέσεων


.text
.
.
.
li $t1,4          #ένας τρόπος για πρόσβαση στο 2ο στοιχείο του πίνακα
lw $t0,vector($t1)
.
.
la $t2,vector           #ακόμα ένας τρόπος για πρόσβαση στο 2ο στοιχείο του
lw $t3,4($t2)           #πίνακα
.
.
.
la $t2,vector           #ακόμα ένας τρόπος για πρόσβαση στο 2ο στοιχείο του
addi $t2,$t2,4          #πίνακα
lw $t0,0($t2)
```

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

1η Ασκηση (25%)

Να γίνει πρόγραμμα σε assembly στο οποίο ο χρήστης θα γεμίζει έναν πίνακα ακεραίων 5 θέσεων. Το πρόγραμμα θα πρέπει να ελέγχει εάν οι τιμές που δίνει ο χρήστης είναι μεγαλύτερες ή ίσες του -21 και μικρότερες ή ίσες του 17. Σε αντίθετη περίπτωση να εμφανίζει κατάλληλο μήνυμα και να ξαναζητάει τον αριθμό. Στη συνέχεια το πρόγραμμα θα υπολογίζει και θα εμφανίζει το άθροισμα των στοιχείων του πίνακα που είναι μεγαλύτεροι του 0 (>0).

2η Ασκηση (25%)

Υλοποιήστε ένα πρόγραμμα σε assembly το οποίο θα βρίσκει τον μέγιστο αριθμό MAX, και τον ελάχιστο αριθμό MIN σε ένα πίνακα ακεραίων καθώς και την θέση τους μέσα στον πίνακα.

Το πρόγραμμα 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. Στην συνέχεια ο χρήστης θα εισάγει από την κονσόλα *n* αριθμούς και θα τους αποθηκεύει σε έναν πίνακα *input* στο τμήμα **.data** της μνήμης.
Στην συνέχεια θα διαπερνάει τον πίνακα και θα βρίσκει τον μεγαλύτερο αριθμό και την θέση του, καθώς και τον μικρότερο αριθμό και την θέση του στον πίνακα. Οι τιμές των αριθμών αυτών και οι θέσεις τους θα εκτυπώνονται στην κονσόλα.
Για παράδειγμα, εάν n=6, και A={23, -23, 4, 0, 1, 11}, τότε
MAX = 23, POS_MAX = 0
MIN = -23, POS_MIN=1

3η Ασκηση (Count number of occurences) (50%)

Υλοποιήστε 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. Στην συνέχεια ο χρήστης θα εισάγει από την κονσόλα *n* αριθμούς με τιμές 0 έως 15 και θα τους αποθηκεύει σε έναν πίνακα Stream στο τμήμα **.data** της μνήμης**.** Εάν ο χρήστης εισάγει έναν αριθμό εκτός του διαστήματος [0..15], το πρόγραμμα θα πρέπει να τον αγνοεί και να ζητάει πάλι την εισαγωγή του.
Το πρόγραμμα θα πρέπει να μετράει πόσες φορές (από τις *n*) εμφανίζεται κάθε αριθμός 0 έως 15. Έστω, για παράδειγμα, $n = 8$, και ο χρήστης εισάγει: 12, 0, 5, 6, 5, 10, 5, 11. Το πρόγραμμα γράφει τις τιμές σε έναν πίνακα Stream[8], ως εξής:
Stream[0] = 12
Stream[1] = 0
Stream[2] = 5
Stream[3] = 6
Stream[4] = 5
Stream[5] = 10
Stream[6] = 5
Stream[7] = 11

Τότε, ο πίνακας OCC [16] (occurencces) στο τμήμα **.data** της μνήμης θα τυπώνεται και είναι:

OCC[0] = 1
OCC[1] = 0
OCC[2] = 0
OCC[3] = 0
OCC[4] = 0
OCC[5] = 3
OCC[6] = 1
OCC[7] = 0
OCC[8] = 0
OCC[9] = 0
OCC[10] = 1
OCC[11] = 1
OCC[12] = 1
OCC[13] = 0
OCC[14] = 0
OCC[15] = 0

Ο πίνακας OCC πρέπει να τυπώνεται στην κονσόλα όπως δείχνουμε στο παράδειγμα. Ο πίνακας Stream δεν χρειάζεται να τυπώνεται.

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