2η Εργαστηριακή Άσκηση

Φόρτωση/Αποθήκευση μεταβλητών και Κλήση Συναρτήσεων

Μέρος Α: Εισαγωγή στις εντολές φόρτωσης και αποθήκευσης

Τα προγράμματα της assembly του MIPS είναι χωρισμένα σε δύο τμήματα, στο τμήμα κειμένου και στο τμήμα δεδομένων. Τα περιεχόμενα του τμήματος κειμένου περιέχουν τις προς εκτέλεση εντολές ενώ τα περιεχόμενα του τμήματος δεδομένων περιέχουν τα δεδομένα που αποθηκεύονται στη μνήμη από το πρόγραμμα και τα οποία μπορούν όχι μόνο να διαβαστούν αλλά και να τροποποιηθούν. Στη συγκεκριμένη άσκηση θα εξοικειωθείτε με τις εντολές φόρτωσης δεδομένων από τη μνήμη καθώς και τις αντίστροφες αυτών, τις εντολές αποθήκευσης δεδομένων στη μνήμη.

Δίνεται ο ακόλουθος κώδικας, τον οποίο καλείστε να μελετήσετε και να περιγράψετε μέσω σχολίων τι ακριβώς κάνει.

```
.data str: .byte 'a','b',-8,'127','e','f','g'

      .text
      .globl main
main:

load: la $t0,mem
      lb $t1,0x0($t0)
      lb $t2,0x1($t0)
      lb $t3,0x2($t0)
      lbu $t4,0x2($t0)
      lb $t5,0x3($t0)
      lbu $t6,0x3($t0)
      lh $t7,0x2($t0)
      lw $t8,0x0($t0)
```

```
store: la $t0, str
       li $t1,'H'
       sb $t1,0x0($t0)
       li $t1,'e'
       sb $t1,0x1($t0)
       li $t1,'l'
       sb $t1,0x2($t0)
       li $t1,'l'
       sb $t1,0x3($t0)
       li $t1,'o'
       sb $t1,0x4($t0)
       li $t1,0
       sb $t1,0x5($t0)
       li $t1,'b'
       sb $t1,0x6($t0)
      li $t1,'o'
       sb $t1,0x7($t0)
      li $t1,'o'
       sb $t1,0x8($t0)

       li $v0,4
       la $a0,str
       syscall

print: li $v0,4
       la $a0,str
       syscall

exit: li $v0,10
       syscall
```

Εργασία

Να γραφεί πρόγραμμα το οποίο να υπολογίζει το άθροισμα 5 αριθμών, που βρίσκονται αποθηκευμένοι στη μνήμη (αρχίζοντας από τη θέση array) και να αποθηκεύει το άθροισμα στη θέση μνήμης sum.

Μέρος Β: Κλήση συναρτήσεων

Στην assembly MIPS έχει γίνει πρόβλεψη για ειδική εντολή για την κλήση διαδικασιών, την *jal (jump-and-link).* Η συγκεκριμένη εντολή εκτελεί άλμα σε μια διεύθυνση και αποθηκεύει κατάλληλα τη διεύθυνση της επόμενης εντολής, ώστε να μπορέσει το πρόγραμμα να συνεχίσει την εκτέλεσή του μετά την επιστροφή από τη διαδικασία. Για το πέρασμα ορισμάτων αλλά και την επιστροφή τιμών από μια διαδικασία, ακολουθείται η ακόλουθη σύμβαση για τους καταχωρητές του MIPS κατά την κλήση διαδικασιών:

- $a0-$a3: Οι συγκεκριμένοι 4 καταχωρητές χρησιμοποιούνται για το πέρασμα των 4 πρώτων ορισμάτων στη διαδικασία.
- $v0-$v1: Οι συγκεκριμένοι 2 καταχωρητές χρησιμοποιούνται για τις δύο πρώτες επιστρεφόμενες τιμές από τη διαδικασία.
- $ra: Στο συγκεκριμένο καταχωρητή αποθηκεύεται η διεύθυνση επιστρο- φής από μια κλήση (εντολή *jal*).
- $t0-$t9: Οι συγκεκριμένοι 10 καταχωρητές χρησιμοποιούνται για προ- σωρινή αποθήκευση και δε διατηρούνται ανάμεσα στις κλήσεις διαδικασιών.
- $s0-$s7: Οι συγκεκριμένοι 8 καταχωρητές χρησιμοποιούνται για μακροχρόνια αποθήκευση και διατηρούνται ανάμεσα στις κλήσεις διαδικασιών.

Για την κλήση μιας διαδικασίας, αρχικά το καλούν πρόγραμμα τοποθετεί τις τιμές των παραμέτρων στους καταχωρητές $a0-$a3. Ακολούθως εκτελεί την εντολή jal ProcAdress, η οποία:

- αποθηκεύει στον καταχωρητή $ra τη διεύθυνση επιστροφής, δηλαδή τη διεύθυνση της επόμενης από τη jal εντολής.
- εκτελεί άλμα στη διεύθυνση της διαδικασίας, δηλαδή στη διεύθυνση ProcAdress.

Στη συνέχεια, η καλούμενη διαδικασία εκτελεί τους όποιους υπολογισμούς και τοποθετεί τα αποτελέσματα στους καταχωρητές $v0 και $v1. Τέλος, επιστρέφει τον έλεγχο στο καλούν πρόγραμμα με την εκτέλεση της εντολής jr $ra, η οποία οδηγεί σε άλμα στη διεύθυνση που βρίσκεται αποθηκευμένη στον καταχωρητή $ra.

Εργασία

Καλείστε να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει από την κονσόλα δύο αριθμούς και θα βρίσκει και θα εκτυπώνει το μεγαλύτερο από τους δύο. Η εύρεση του μεγίστου θα γίνεται από μια ξεχωριστή διαδικασία, my_max. Ακολούθως δίνεται ένας σκελετός του προγράμματος:

```
.data
str1: .asciiz "Dwse ton prwto arithmo: "
str2: .asciiz "Dwse to deutero arithmo: "
str3: .asciiz "O megalyteros arithmos einai:"
nl: .asciiz "\n"
.text
.globl main
main:
li $v0,4 # print string
la $a0,str1
syscall

li $v0,5 # read int
syscall
move $s0,$v0

li $v0,4 # print string
la $a0,str2
syscall

li $v0,5 # read int
syscall
move $s1,$v0

li $v0,4 # print string
la $a0,str3
syscall

# Insert your Code Here

jal my_max

# Insert your Code Here

li $v0,1 # print int
syscall

li $v0,4 # print string
la $a0,nl
syscall

li $v0,10 # exit
syscall

my_max:

# Insert your Code Here

jr $ra
```