# 3ο Εργαστήριο Αρχιτεκτονικής Η/Υ: MIPS assembly: Μετατροπή string σε αριθμό

Α. Ευθυμίου

**Παραδοτέο: Παρασκευή 13 Μάρτη, 23:00**

Το αντικείμενο αυτής της άσκησης είναι ένα πρόγραμμα που μετατρέπει ένα string που περιέχει έναν δεκαεξαδικό αριθμό στον αριθμό με τη συνηθισμένη δυαδική αναπαράσταση. Σε αυτή την εργαστηριακή άσκηση θα δείτε και θα γράψετε προγράμματα assembly που χρησιμοποιούν εντολές ολίσθησης, λογικές πράξεις, συγκρίσεις και διακλαδόσεις. Θα πρέπει να έχετε μελετήσει τα μαθήματα για τη γλώσσα assembly του MIPS που αντιστοιχούν μέχρι την ενότητα 2.7 και, επίσης, την ενότητα 2.9 του βιβλίου.

Σας δίνεται ένα πρόγραμμα που κάνει ακριβώς το ανάποδο: μετατρέπει έναν αριθμό σε ένα string που θα μπορούσε να χρησιμοποιηθεί για να προβληθεί ο αριθμός σε δεκαεξαδική μορφή.

## 1 Αριθμοί και strings

Μέχρι τώρα έχετε δεί πολλές φορές τον τρόπο με τον οποίο αναπαριστώνται οι ακέραιοι αριθμοί σε έναν υπολογιστή. Όμως όταν προβάλουμε έναν αριθμό στην οθόνη, πρέπει να εμφανίζουμε το κάθε ψηφίο του χωριστά, μια και η οθόνη περιμένει ένα χαρακτήρα ανά ψηφίο, συνήθως σε κωδικοποίηση ASCII[1]. Αντίστροφα, κατά την είσοδο αριθμών από το πληκτρολόγιο, δίνουμε μια σειρά από ψηφία τα οποία τελικά μετατρέπονται σε έναν (δυαδικό) αριθμό. Είναι λοιπόν πολύ συνηθισμένο να μετατρέπουμε αριθμούς σε strings και το αντίστροφο.

Σε αυτή την άσκηση θα περιοριστούμε σε ακέραιους αριθμούς 32 bit, χωρίς πρόσημο και τα strings θα παριστάνουν τον αριθμό σε δεκαεξαδική μορφή. Αφού λοιπόν έχουμε 32bit αριθμούς, τα string θα είναι το πολύ 8 δεκαεξαδικών ψηφίων (0-9 και A-F). Επίσης τα strings που θα διαβάζουμε ή θα γράφουμε θα τελειώνουν με τον ειδικό χαρακτήρα `'\0'`, που κωδικοποιείται ως 0 στον κώδικα ASCII.

Τέλος δεν θα ελέγχουμε αν υπάρχει κάποιος λάθος χαρακτήρας (διαφορετικός από 0-9 και A-F) σε ένα string, και δεν θα χρησιμοποιούμε το σύνηθες πρόθεμα 0x που δηλώνει ότι ακολουθεί δεκαεξαδικός αριθμός.

## 2 Το πρόγραμμα i2hex.asm

Σας δίνεται το πρόγραμμα i2hex.asm που μετατρέπει αριθμούς σε string. Δέχεται έναν αριθμό 32bit, αποθηκευμένο στη μνήμη (label number) και τη διεύθυνση στην οποία θα αποθηκευτεί το string (label outmsg), όπου και γράφει το αποτέλεσμα.

Θα το βρείτε στο GitHub και μπορείτε να το πάρετε, μαζί με τα υπόλοιπα αρχεία της άσκησης 3 δίνοντας τις εξής εντολές:

```
git remote add lab03_starter https://github.com/UoI-CSE-MYY402/lab03_starter.git
git fetch lab03_starter
git merge lab03_starter/master -m "Fetched lab03 starter files"
```

Μελετήστε καλά το πρόγραμμα i2hex.asm στον κατάλογο lab03 που θα εμφανιστεί. Ένας καλός τρόπος να καταλάβετε τι κάνει είναι να το τρέξετε στο MARS και να παρατηρήσετε προσεκτικά τις τιμές των καταχωρητών μετά από κάθε εντολή assembly που εκτελείται.

Στην αρχή, στο τμήμα δεδομένων, παρατηρήστε ένα καινούριο assembly directive το `.space` που κρατάει έναν αριθμό από bytes στη μνήμη. Στο i2hex.asm κρατούνται 9 bytes, χώρος αρκετός για 8 χαρακτήρες - δεκαεξαδικά ψηφία και το `'\0'`.

Η μετατροπή γίνεται χωρίζοντας τον 32bit αριθμό σε ομάδες των 4 bit, που αντιστοιχούν σε ένα δεκαεξαδικό ψηφίο. Ο κάθε αριθμός των 4 bit (στον καταχωρητή `$t1`) μετατρέπεται σε ένα χαρακτήρα με ένα

[1] Ψάξτε την κωδικοποίηση ASCII αν δεν τη γνωρίζετε.

τέχνασμα. Επειδή οι χαρακτήρες '0' ώς '9' αναπαρίστανται ώς συνεχόμενοι αριθμοί στην κωδικοποίηση ASCII (από το $48_{ten}$ ως το $57_{ten}$), προσθέτουμε τον 4 bit αριθμό στο 48 για να πάρουμε τον κωδικό του αντίστοιχου χαρακτήρα. Δυστυχώς οι χαρακτήρες 'A' ώς 'F', αν και συνεχόμενοι και αυτοί, δεν είναι αμέσως μετά το '9' και επομένως χρειάζεται να κάνουμε μια σύγκριση για να γνωρίζουμε αν ο 4 bit αριθμός είναι μέχρι το 9 ή μεγαλύτερος.

Προσέξτε πως φαίνεται το τελικό αποτέλεσμα ως string στη μνήμη (επιλέξτε το ASCII κάτω δεξιά στο παράθυρο Data segment). Αν η σειρά των ψηφίων σας φαίνεται κάπως ανάποδη, σκεφτείτε το λίγο περισσότερο. Η σειρά είναι σωστή!

Όταν είστε βέβαιοι ότι καταλάβατε καλά πως δουλεύει, προχωρήστε στο επόμενο βήμα και γράψτε ένα πρόγραμμα που κάνει το ανάποδο.

## 3 Μετατροπή string σε αριθμό, hex2i

Έχοντας το i2hex ως οδηγό, γράψτε το πρόγραμμα hex2i που μετατρέπει ένα string, το οποίο αναπαριστά έναν δεκαεξαδικό αριθμό με το πολύ 8 ψηφία, στον αντίστοιχο 32bit αριθμό. Χρησιμοποιείστε τον σκελετό που βρίσκεται στο αρχείο hex2i.asm. Προσέξτε ότι το τελικό αποτέλεσμα πρέπει να βρίσκεται στον καταχωρητή `$s0`.

Προσοχή, ενώ το i2hex παράγει πάντα strings με 8 ψηφία, συμπληρώνοντας με 0 στα αριστερά αν χρειάζεται, το hex2i θα πρέπει να είναι σε θέση να διαβάσει έναν αριθμό με λιγότερα από 8 ψηφία! Θα μπορείτε πάντως να υποθέσετε ότι θα σας δίνεται τουλάχιστον 1 ψηφίο και ποτέ πάνω από 8 ψηφία. Γενικά μπορείτε να υποθέσετε ότι η είσοδος θα είναι σωστή (μόνο δεκαεξαδικά ψηφία και θα τελειώνει με `'\0'` ) και δεν χρειάζεται να την ελέγχετε.

Χρησιμοποιείστε ολισθήσεις αντί για πολλαπλασιασμό. (Απαγορεύεται να χρησιμοποιήσετε πολλαπλασιασμό!) Προσπαθείστε, επίσης, να αποφύγετε την πρόσθεση για να «συνθέσετε / κολλήσετε» τα επιμέρους ψηφία - τετράδες από bit στον αριθμό. Μπορείτε να χρησιμοποιήσετε μια λογική πράξη για την «σύνθεση».

## 4 Παραδοτέο

Το παραδοτέο της άσκησης είναι το αρχείο hex2i.asm που περιέχει το πρόγραμμά σας.

Πρέπει να κάνετε commit τις αλλαγές σας και να τις στείλετε (push) στο GitHub repository για να βαθμολογηθούν πριν από την καταληκτική ημερομηνία!

Το πρόγραμμά σας θα βαθμολογηθεί για την ορθότητά του, την ποιότητα σχολίων και την ταχύτητα εκτέλεσής του.