ΨΗΦΙΑΚΑ ΥΠΟΛΟΓΙΣΤΙΚΑ ΣΥΣΤΗΜΑΤΑ - HPY 202
ΦΥΛΛΟ ΑΝΑΦΟΡΑΣ ΓΙΑ ΤΙΣ ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ 3,4,5

Αριθμός ομάδας: 4
Ονοματεπώνυμα Μελών: Γεώργιος Πετακάκης

Σκοπός των ασκήσεων

Ο σκοπός εκπόνησης των τριών ασκήσεων, ήταν για την μάθηση ανάπτυξης κώδικα με την assembly γλώσσα του RISC επεξεργαστή MIPS, που πέρα από την βαθιά ιστορία του, πολλά ενσωματωμένα συστήματα τον χρησιμοποιούν και κατανοώντας τον τρόπο λειτουργίας του, ταυτοχρόνως έχουμε και μία βαθύτερη κατανόηση του τρόπου λειτουργίας των συστημάτων αυτών.

Περιγραφή Επίλυσης των ασκήσεων

Μία συνάρτηση έχει παραμέτρους και για αυτό χρησιμοποιούνται οι καταχωρητές ($a0,$a1,...) , μετά για τοπικές μεταβλητές ($s0,s1,...) όπου δεσμεύουμε και αποδεσμεύουμε στην stack και για return value ($v0,...). Η δόμησή της έχει τρία μέροι, τον πρόλογο, το κυρίως θέμα και τον επίλογο. Στο εργαστήριο 3 δεν το κάναμε με τέτοια λογική, στο εργαστήριο 4 δεν κάναμε σωστά τον πρόλογο και στο στο εργαστήριο 5 δεν χρησιμοποιήσαμε παραμέτρους και return value.

Στο εργαστήριο 3:

Για loops χρησιμοποιήσαμε jump, για κλήση σε συναρτήσεις μίας χρήσης ανά loop χρησιμοποιήσαμε jump and link (jal). Το μόνο μέρος που έβαλα jump ήταν στην κάθαρση των χαρακτήρων από τους control όταν δίνεται ο (@), γιατί δεν επιστρέφεται το πρόγραμμα στο να διαβάζει χαρακτήρες.

Στο εργαστήριο 4:

Υλοποιήσαμε την κλήση των συναρτήσεων στην main, σε αυτό το εργαστήριο κάναμε και την χρήση της stack για καταχωρητές τύπου ($s) αν και όχι πλήρως σωστή. Στο διάβασμα των χαρακτήρων βάλαμε jump για δημιουργία loop και όταν σε έναν από τους καταχωρητές διαβάζεται ο (@) με μία εντολή branch-if equal. Πάει σε μια υπορουτίνα της όπου αποδεσμεύει τους καταχωρητές μέσα στην stack και μετά κάνει jr στην main όπου και την κάλεσε. Το ίδιο γίνεται και στην συνάρτηση κάθαρσης χαρακτήρων.

Στο εργαστήριο 5:

Σε αυτο το εργαστήριο χρησιμοποιήθηκε ένα “main menu” όπου ανάλογα στην επιλογή του χρήστη, οδηγείται και στον ανάλογο μέρος του κώδικα με εντολές branch-if equal. Για την επιλογή (E) διαβάζει τα τρία ανάλογα strings που δεν είναι όλα μέσα σε ένα label αλλά τρία διαφορετικά και ενωμένα με την jump. Στην αποθήκευση όπου έπρεπε να αφαιρεθεί το (\n) πριν την αποθήκευση σε μέρος μνήμης του καταλόγου, κάναμε nested-function με την jal,

διότι θέλαμε να γυρίσει στην caller function, μετά την επεξεργασία του string και φυσικά ήξερε που να γυρίσει διότι αποθηκεύτηκε το return adress στην stack (αναδρομή). Μετά την ολοκλήρωση της διαδικασίας κάνει jump στην main , όπου και ρωτάει τον χρήστη για την επόμενη κίνηση.

Συμπεράσματα:

Όλοι η διαδικασία των ασκήσεων της assembly, ήταν διασκεδαστική, διότι για πρώτη φορά, δόθηκε η ελευθερία να γράψουμε κώδικα χωρίς να βασιστούμε σε πλήρες λογική και τις συμβάσεις άλλων ανθρώπων, πέρα από την δική μας (πέρα από την μάθηση κάθε τύπου καταχωρητή και τις διαφορετικές χρήσης του με βάση της συμβάσεις των δημιουργών). Η assembly είναι μία γλώσσα που ξεχωρίζει απίστευτα σε σχέση με κάθε άλλη και όσοι ασχολήθηκαν, είναι πιθανό ότι έχουν την ίδια άποψη.