Δεύτερο Εργαστήριο RISC-V

Εργαστήριο Μικροϋπολογιστών

Σε αυτό το εργαστήριο θα προγραμματίσετε τα LEDs της πλατφόρμας RVfpga γράφοντας απευθείας κώδικα assembly. Αυτή η βασική διαφορά σε σχέση με την πρώτη άσκηση (η οποία υλοποιήθηκε σε γλώσσα C) θα επιφέρει μεγαλύτερη εξοικείωση με το ISA RV32I, και μπορεί να αποτελέσει τη βάση για περαιτέρω μελλοντική σας ενασχόληση με τη συγκεκριμένη οικογένεια επεξεργαστών.

Αρχικοποίηση περιβάλλοντος

Το περιβάλλον προγραμματισμού και debugging παραμένει ίδιο με αυτό που χρησιμοποιήσατε για τις ανάγκες της πρώτης άσκησης. Για να ξεκινήσετε τη γραφή του κώδικά σας, ακολουθήστε τα ακόλουθα βήματα:

1. Δημιουργήστε ένα νέο PlatformIO project.
2. Τροποποιήστε κατά τα γνωστά το αρχείο platformio.ini.
3. Δημιουργήστε στον φάκελο src του project σας ένα νέο αρχείο με όνομα της επιλογής σας και κατάληξη .S (παράδειγμα: dummy.S).

Αρχικοποίηση αρχείου assembly

Το ISA RISC-V παρέχει στον προγραμματιστή μια ποικιλία ειδικών οδηγιών, ονόματι directives, που διευκολύνουν (αν όχι επιτρέπουν) τη γραφή λειτουργικού και ουσιώδους κώδικα. Για παράδειγμα, το directive .globl ενημερώνει τον compiler πως η αντίστοιχη ετικέτα-διεύθυνση μπορεί να αναφερθεί και από αρχεία διαφορετικά του παρόντος (όπως έχουμε δει, το firmware του RVfpga πρώτα θα κάνει διάφορες δικές του ενέργειες και κατόπιν θα πηδήξει στη main).

Αντίστοιχα directives υπάρχουν για τη δήλωση σταθερών, αρχικοποίηση διανυσμάτων, κτλ. Άλλα από αυτά είναι απαραίτητα να συμπεριληφθούν, και άλλα προαιρετικά. Ο **Πίνακας 1** συνοψίζει όλα τα directives και τις περιγραφές τους.

Το ακόλουθο παράδειγμα αποτελείται από την αρχή ενός προγράμματος που επεξεργάζεται δύο μονοδιάστατους πίνακες μήκους 10 στοιχείων για να παράξει έναν τρίτο, ισομήκη πίνακα. Συναρτήσει του Πίνακα 1, είναι εύκολο να εξάγετε συμπεράσματα για το πού συνεισφέρει το κάθε directive. Κάποια ενδιαφέροντα σχόλια που μπορούν να γίνουν είναι:

- γιατί ενώ ο πίνακας C είναι 10 στοιχείων (όσων δηλαδή και οι A, B) δεσμεύουμε για αυτόν μνήμη τετραπλάσιου μεγέθους;
    - διότι από το τμήμα κάτω από το directive .data βλέπουμε ότι τα στοιχεία των πινάκων είναι τύπου **word** (μήκος 4 bytes, byte addressed μνήμη)
- ποιοι καταχωρητές χρησιμοποιούνται στο σώμα της main;

- κατά κόρον οι temporary καταχωρητές t0, t1 κ.ο.κ.

- ποια από τα directives του παραδείγματος φαίνονται απαραίτητα για τη γραφή της πλειοψηφίας απλοϊκών προγραμμάτων σαν και αυτό;
  - hint: **δεν** είναι το .equ

```
.globl main                                    #

.equ N, 10

.data
A: .word 0,1,2,7,-8,4,5,-12,11,-2
B: .word 0,1,2,7,-8,4,5,12,-11,-2

.bss
C: .space 4*N

.text
main:
   la t0, A
   la t1, B
   add t1, t1, 4*(N-1)
#Rest of the program follows . . .
#...
.end
```

Πίνακας 1: RISC-V directives.

Directive	Description
`.text`	Subsequent items are stored in the `text` section (machine code).
`.data`	Subsequent items are stored in the `data` section (global variables).
`.bss`	Subsequent items are stored in the `bss` section (global variables initialized to 0).
`.section .foo`	Subsequent items are stored in the section named `.foo`.
`.align n`	Align the next datum on a 2^n-byte boundary. For example, `.align 2` aligns the next value on a word boundary.
`.balign n`	Align the next datum on an n-byte boundary. For example, `.balign 4` aligns the next value on a word boundary.
`.globl sym`	Declare that label `sym` is global and may be referenced from other files
`.string "str"`	Store the string `str` in memory and null-terminate it.
`.word w1,…,wn`	Store the `n` 32-bit quantities in successive memory words.
`.byte b1,…,bn`	Store the `n` 8-bit quantities in successive bytes of memory.
`.space`	Reserve memory space to store variables without an initial value. It is commonly used to declare the output variables, when they are not also serving as input variables. The space we want to reserve must always be expressed as a number of bytes. For example, the directive `RES: .space 4` reserves four bytes (i.e. one word) that are not initialized.
`.equ name,constant`	Define symbol `name` with value `constant`. For example, `.equ N,12`, defines symbol N with the value 12.
`.end`	The assembler will conclude its work when it reaches the directive `.end`. Any text located after this directive will be ignored.

- πώς μπορώ να εποπτεύσω τις θέσεις μνήμης που παρέχονται σε κάθε έναν από τους πίνακες, εφ' όσον αυτές δεν αναφέρονται ρητά;
    - αφήνεται ως άσκηση

Ζητούμενα

Οι παραπάνω πληροφορίες επαρκούν για την υλοποίηση των ακόλουθων ερωτημάτων. Όπως και στην πρώτη άσκηση, *υλοποιήστε κάθε ερώτημα ως ξεχωριστό PlatformIO project.* Αντίθετα με την πρώτη άσκηση, τα προγράμματά σας δεν είναι απαραίτητο να είναι συνεχούς λειτουργίας.

Ερώτημα 1

Συμπληρώστε τον κώδικα του παραδείγματος ώστε στον πίνακα C να αποθηκεύονται όλα τα στοιχεία της μορφής:

$$C(i) = |A[i] + B[N-i-1]|,\ i = 0,..,N-1.$$

Ερώτημα 2

Να γραφεί κώδικας assembly που εκτελεί την εξής σειρά βημάτων:

1. Άναμμα λιγότερο σημαντικού LED
2. "Ολίσθηση" ένδειξης προς τα αριστερά (δηλαδή σβήσιμο λιγότερο σημαντικού LED και άναμμα 2ου λιγότερο σημαντικού)
3. Επανάληψη βήματος 2 μέχρις ότου η ένδειξη φτάσει το πιο σημαντικό LED
4. Κρατώντας το πιο σημαντικό LED αναμμένο, επανάληψη βήματος 1
5. Επανάληψη βήματος 2 μέχρις ότου η ένδειξη φτάσει το **2ο** πιο σημαντικό LED
6. Κρατώντας τα 2 πιο σημαντικά LED αναμμένα, επανάληψη βήματος 1

Όταν ανάψουν όλα τα LEDs, φροντίστε να ξεκινάει το **σβήσιμό** τους ακολουθώντας την ακριβώς αντίστροφη διαδικασία. Δηλαδή:

1. Σβήσιμο σημαντικότερου LED
2. "Ολίσθηση" μη-ένδειξης προς τα δεξιά (δηλαδή άναμμα σημαντικότερου LED και σβήσιμο 2ου σημαντικότερου)
3. Κ.ο.κ. .....

Παραδοτέα

Η επιτυχής εξέταση της παρούσας άσκησης μεταφράζεται **μόνο** ως εμπρόθεσμη παράδοση αναλυτικής αναφοράς, συνοδευόμενης από τα αντίστοιχα PlatformIO projects. **Δεν** θα πραγματοποιηθεί προφορική εξέταση, όμως οι κώδικές σας θα βαθμολογηθούν

αυστηρά (δώστε μεγάλη έμφαση, όπως και στην προηγούμενη άσκηση, στον σχολιασμό του κώδικά σας).

Ως τελική προθεσμία για την παράδοση της άσκησης ορίζεται η **Παρασκευή 22/1**.