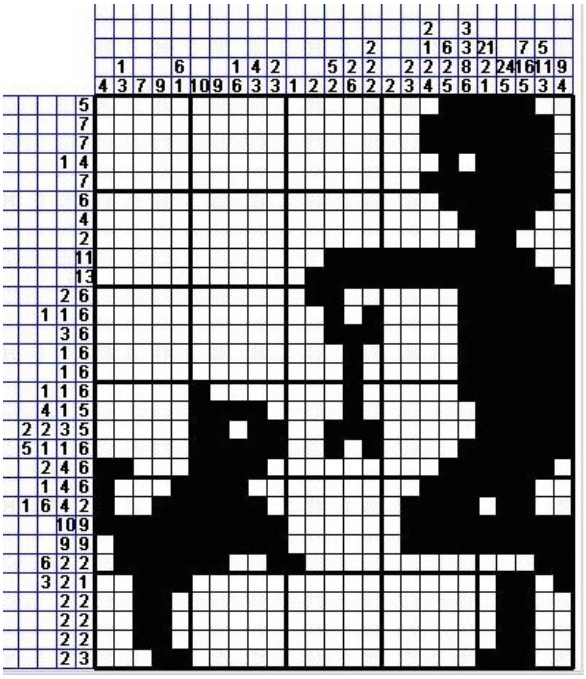
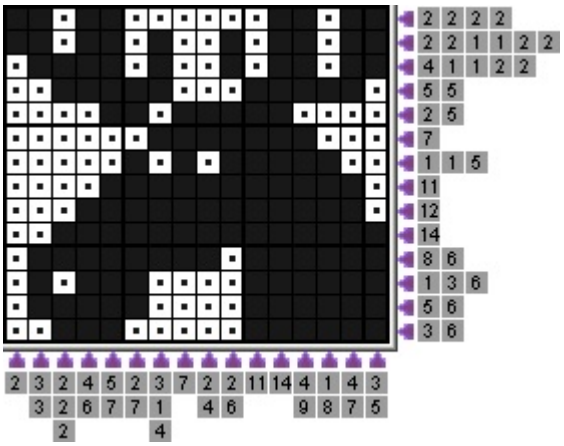
# 3ο Εργαστήριο Αρχιτεκτονικής Η/Υ: MIPS assembly: Nonograms

Α. Ευθυμίου

**Παραδοτέο: Τρίτη 8 Μάρτη, 23:00**



Σχήμα 1: Dog



Σχήμα 2: Moose

Το αντικείμενο αυτής της άσκησης είναι ένα πρόγραμμα που επεξεργάζεται μία εικόνα σχηματισμένη από ασπρόμαυρες κουκίδες, και υπολογίζει τους αριθμούς - στοιχεία που χρειάζονται για να μετατραπεί η εικόνα σε ένα Γιαπωνέζικο γρίφο nonogram. Θα πρέπει να έχετε μελετήσει τα μαθήματα για τη γλώσσα assembly του MIPS που αντιστοιχούν μέχρι την ενότητα 2.7 του βιβλίου (4η διάλεξη).

## 1 Nonograms

Τα nonogram, επίσης γνωστά ως Hanjie ή Picross, είναι γρίφοι λογικής. Έχουν τη μορφή ενός ορθογώνιου πίνακα από τετραγωνάκια τα οποία πρέπει να χρωματιστούν ή να μείνουν λευκά όπως ορίζουν αριθμοί - στοιχεία (clues) που βρίσκονται σε κάθε σειρά και στήλη. Οι αριθμοί καθορίζουν τον αριθμό των συνεχόμενων τετραγώνων που πρέπει να χρωματιστούν στη συγκεκριμένη γραμμή ή στήλη. Για παράδειγμα αν μια γραμμή έχει τους αριθμούς 3 7 2, αυτό σημαίνει ότι θα υπάρχουν 3, 7 και 2 συνεχόμενα χρωματιστά τετράγωνα, *με αυτή τη σειρά* από αριστερά προς τα δεξιά και *με τουλάχιστον ένα κενό τετράγωνο ανάμεσά τους*. (Για αριθμους στηλών η σειρά είναι από επάνω προς τα κάτω.)

Ο λύτης στην ουσία πρέπει να βρεί την κρυμένη εικόνα λύνοντας τον γρίφο. Συχνά υπάρχουν μόνο 2 χρώματα (άσπρο-μαύρο), αλλά υπάρχουν και πολύχρωμα nonogram. Για την άσκηση, το πρόγραμμα που θα γράψετε δεν θα λύνει nonograms, αλλά θα παίρνει μια αναπαράσταση εικόνας και θα υπολογίζει τους αριθμούς-στοιχεία των γραμμών και των στηλών.

## 2 Πληροφορίες για την άσκηση

Για να πάρετε το σκελετό της εργαστηριακής άσκησης, μεταβείτε στον κατάλογο εργασίας που είχατε κλωνοποιήσει από το αποθετήριό σας στο GitHub. (cd <όνομα χρήστη GitHub>-labs). Μετά, κάνετε τα

παρακάτω βήματα:

```
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"
```

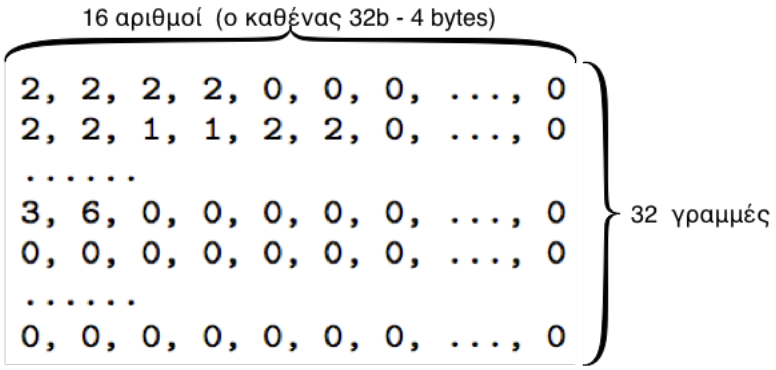
Το αρχείο lab03.asm περιλαμβάνει μόνο το τελικό syscall τερματισμού του προγράμματος και τα δεδομένα εισόδου για το nonogram του σχήματος 1.

Τα nonograms που θα μπορεί να επεξεργαστεί το πρόγραμμα θα έχουν ως ελάχιστη διάσταση το 1 τετραγωνάκι και ως μέγιστη τα 32 τετραγωνάκια, αλλά οι διαστάσεις δεν θα είναι απαραίτητα ίσες. Η εικόνα που θα δίνεται ως είσοδος θα παριστάνεται ως ένας πίνακας 32bit αριθμών, ένας για κάθε σειρά από επάνω προς τα κάτω. Κάθε αριθμός του πίνακα εικόνας κωδικοποιεί τα τετραγωνάκια που είναι μαυρισμένα σε αυτή τη γραμμή θέτοντας το bit που αντιστοιχεί στο τετραγωνάκι στην τιμή 1. Όταν ο αριθμός των στηλών είναι μικρότερος από 32, χρησιμοποιούνται μόνο τα λιγότερο σημαντικά bit. Για παράδειγμα, για το nonogram του σχήματος 2, ο αριθμός που κωδικοποιεί την πρώτη γραμμή είναι 0x0d81b (δυαδικός χωρισμένος σε 4δες: 0000_1101_1000_0001_1011).

Στο lab03.asm, η εικόνα είναι αποθηκευμένη σαν πίνακας στην ετικέτα picture και ο αριθμός των γραμμών και στηλών στις ετικέτες rows, cols αντίστοιχα.

Οι αριθμοί-στοιχεία κάθε γραμμής και στήλης έχουν οργανωθεί ως πίνακες δύο διαστάσεων. Επειδή η μέγιστη διάσταση του γρίφου είναι 32, στη χειρότερη περίπτωση μπορεί να υπάρχουν 16 αριθμοί σε μία σειρά ή σε μία στήλη: κάθε άλλο τετραγωνάκι είναι μαυρισμένο, άρα οι αριθμοί είναι δεκαέξι άσοι. Έτσι θεωρούμε ότι οι αριθμοί σειρών (**r_clues**) είναι οργανωμένοι ως ένας πίνακας ακεραίων 32 bit[1] διαστάσεων 32x16 (γραμμές x στήλες). Δηλαδή για κάθε μία από τις (έως) 32 γραμμές, θα υπάρχουν μέχρι 16 αριθμοί-στοιχεία.

Για το nonogram του σχήματος 2, το r_clues θα είναι όπως στο σχήμα 3. Οι τελευταίες γραμμές του πίνακα είναι όλες 0 γιατί το nonogram αυτό έχει 14 γραμμές. Παρόμοια οι τελευταίες στήλες είναι όλες 0.

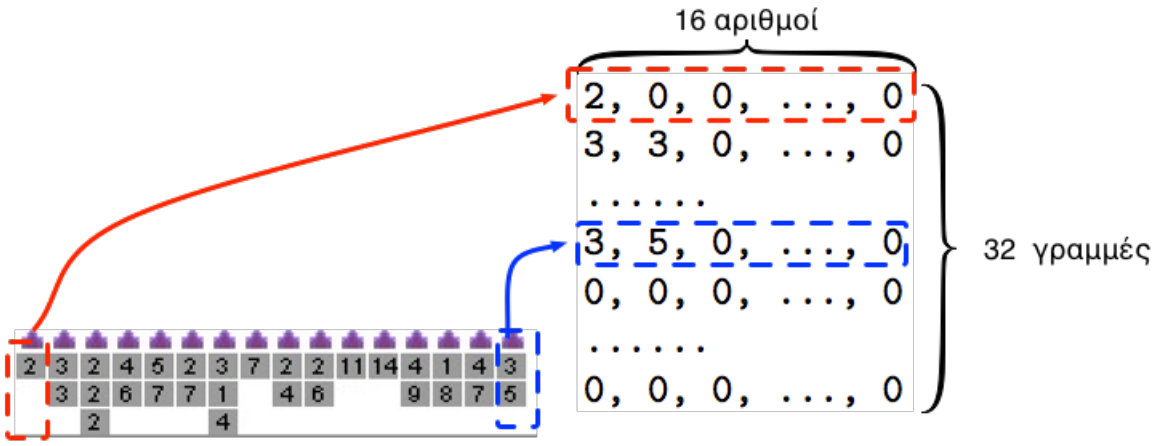



Σχήμα 3: Πίνακας r_clues

Παρόμοια οι αριθμοί στηλών (c_clues) είναι οργανωμένοι ως ένας πίνακας διαστάσεων 32x16. Προσοχή, ενώ ο πίνακας r_clues μοιάζει πολύ στην εικόνα των αριθμών στοιχείων στο πλάι του nonogram, ο πίνακας των στηλών είναι διαφορετικός: οι στήλες των αριθμών στο nonogram είναι πλέον γραμμές στο c_clues όπως φαίνεται στο σχήμα 4 παρακάτω.

Επειδή οι πίνακες θα πρέπει τελικά να αποθηκευθούν στην μνήμη του υπολογιστή που έχει μία μόνο διάσταση, θεωρούμε ότι αποθηκεύονται με την λεγόμενη row-major διάταξη (order): πρώτα αποθηκεύονται τα στοιχεία της πρώτης γραμμής από αριστερά προς τα δεξιά, μετά της δεύτερης, κ.ο.κ. Προσοχή

[1] Θα μπορούσαμε να χρησιμοποιήσουμε bytes αντί για 32bit ακέραιους για οικονομία μνήμης.




Σχήμα 4: Πίνακας c_clues

τα μηδενικά δεν παραλείπονται: κάθε γραμμή του πίνακα αποτελείται από 16 αριθμούς (των 32bit ο καθένας).

Σημαντικό μέρος της επίλυσης της άσκησης είναι να υπολογίσετε τις διευθύνσεις στις οποίες πρέπει να αποθηκευτούν οι αριθμοί, στοιχεία γραμμών και στηλών. Η διεύθυνση του `A[i][j]`, γνωρίζοντας ότι κάθε στοιχείο του πίνακα είναι b bytes, ότι το πρώτο στοιχείο (`A[0][0]`) βρίσκεται στη διεύθυνση A και ότι ο αριθμός των στηλών είναι c, είναι $A + icb + jb$ όταν ο πίνακας είναι οργανωμένος ως row-major. Δοκιμάστε το: για i=j=0, η διεύθυνση, σύμφωνα με την παραπάνω έκφραση, είναι A, για i=0, j=1, η διεύθυνση είναι $A + b$, και για i=1, j=0, η διεύθυνση είναι $A + cb$.

Στην άσκηση αυτή μην χρησιμοποιήσετε την εντολή πολλαπλασιασμού γι'αυτούς τους υπολογισμούς. Μπορούν να γίνουν με ολισθήσεις ή με προσθέσεις κατάλληλα (προ-)υπολογισμένων σταθερών (τα $c, b$ είναι σταθερές).

Παρατηρήστε επίσης ότι στη δήλωση των ετικετών r_clues, c_clues χρησιμοποιείται η οδηγία `.word 0 : 512` που αρχικοποιεί τη μνήμη με την τιμή 0, για 512 (=32x16) λέξεις των 32 bit. Επομένως το πρόγραμμά σας μπορεί να γράφει μόνο τους αριθμούς, στοιχεία που χρειάζονται γιατί οι υπόλοιποι θα είναι ήδη 0.

Ο πιο απλός τρόπος υπολογισμού των αριθμών-στοιχείων είναι με 2 φωλιασμένες επαναλήψεις:
μία που διατρέχει όλες τις γραμμές και για κάθε γραμμή εξετάζει όλα τα τετραγωνάκια υπολογίζοντας τους αριθμούς-στοιχεία της γραμμής και
μία που διατρέχει όλες τις στήλες και για κάθε στήλη εξετάζει ένα-ένα τα τετραγωνάκια, της στήλης αυτής, για κάθε γραμμή.
Σε ψευτοκώδικα:

```
for i = 0 to rows do
    for j = 0 to cols do
        calculate r_clues[i]  // row clues for fow i
for j = 0 to cols do
    for i = 0 to rows do
        calculate c_clues[j]  // column clues for column j
```

Θα χρειαστεί να κάνετε λογικές πράξεις (ολισθήσεις, μάσκες) για να απομονώσετε τις τιμές των bit που αντιστοιχούν στα τετραγωνάκια. Ίσως να χρειαστείτε τις εντολές μεταβλητής ολίσθησης (sllv srlv) που είναι αντίστοιχες με τις εντολές ολίσθησης που είδαμε στο μάθημα, μόνο που αντί να ολισθαίνουν κατά μια σταθερή τιμή, η ολίσθηση καθορίζεται από την τιμή ενός καταχωρητή.

Επειδή το πρόγραμμα είναι αρκετά πολύπλοκο θα χρειαστείτε αρκετές μεταβλητές-καταχωρητές. Είναι

εύκολο να μπερδευτεί κανείς και να κάνει λάθη με τα ονόματα καταχωρητών γιατί μοιάζουν μεταξύ τους και τα ονόματα καταχωρητών δεν θυμίζουν το σκοπό της μεταβλητής που αποθηκεύουν. Γι'αυτό προτείνω στην αρχή να χρησιμοποιείτε μεταβλητές, με κατάλληλα ονόματα, αντί για καταχωρητές. Δηλαδή στην αρχή να γράφετε ένα είδος ψευτοκώδικα με κανονικές εντολές assembly αλλά με μεταβλητές αντί για καταχωρητές. Όταν τελειώσετε τη συγγραφή του προγράμματος, αντικαταστείστε τις μεταβλητές με καταχωρητές, κρατώντας σε σχόλια την αντιστοιχία (π.χ. `$s0` - αρχική διεύθυνση πίνακα picture).

Χρησιμοποιείστε τους καταχωρητές `$s0-7` για μεταβλητές που χρειάζονται για μεγάλα τμήματα του προγράμματος και τους `$t0-9` για τιμές που είναι σχετικά παροδικές, π.χ. αποτέλεσμα μιας σύγκρισης που χρειάζεται για μια διακλάδωση μόνο. Αν δεν φτάνουν οι `$s0-7` χρησιμοποιείστε και άλλους καταχωρητές, εκτός βέβαια από τους `$sp, $at, $k0-1, $ra`.

## 3 Παραδοτέο

Το παραδοτέο της άσκησης είναι το αρχείο lab03.asm που περιέχει το πρόγραμμά σας. Στον κατάλογο test το lab03_tester.py περιέχει τα δεδομένα και πλήρη έλεγχο για τα 2 nonogram αυτού του φυλαδίου και ένα μικρό 1x1 nonogram. Δεν χρειάζεται να κάνετε αλλαγές σε αυτά, παρά μόνο να επιβεβαιώσετε ότι το πρόγραμμά σας περνάει τον έλεγχο.

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 συνοπτικότητά του.