Εργαστήριο 10

23/12/2020

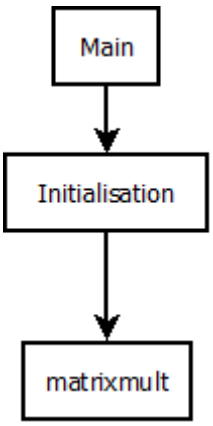
Κάνοντας υπολογισμούς με «μικρούς» μικροελεγκτές – πολλαπλασιασμός 3X3 πινάκων float και int

Ντουνέτας Δημήτρης
AM: 2016030141

Εισαγωγή

Σκοπός του εργαστηρίου είναι η δημιουργία ενός προγράμματος για υπολογισμό πινάκων 3X3 με χρήση int και float αριθμών. Σε αυτό εργαστήριο θέλουμε να παρατηρήσουμε την διαφορά σε πράξεις με αυτά τα 2 είδη αριθμών και πως ανταποκρίνεται διαφορετικά ο μικροελεκτής μας και ο avr gcc Compiler.

Block Diagram Προγράμματος



Αρχικοποίηση του προγράμματος σε Γλώσσα C

Αρχικά το πρόγραμμα ξεκινάει από την συνάρτηση main που βρίσκεται στο αρχείο main.c . Η main συνάρτηση είναι υπεύθυνη για την αρχικοποίηση του προγράμματος και της μνήμης. Η συνάρτηση main καλεί την initialization μέσα στην οποία αρχικοποιούμε 3 θέσεις μνήμης για του πίνακες που θα πολλαπλασιάσουμε και τον πίνακα των αποτελεσμάτων. Ο Stack Pointer δεν χρειάζεται να αρχικοποιηθεί αφού τον αρχικοποιεί ο C Compiler από μόνος του ώστε να μπορούμε να επιστρέφουμε σωστά από τις ρουτίνες και τις συναρτήσεις που καλούμε.

```
volatile int *matrix1 = (int *)0x0070;
volatile int *matrix2 = (int *)0x0082;
volatile int *output = (int *)0x0094;

void initialisation(){
    for (uint8_t i = 0; i<9; i++)
    {
        matrix1[i] = i;
    }
    for (uint8_t i = 0;i<9;i++)
    {
        matrix2[i] = i;
    }
    for(uint8_t i=0;i<9;i++)
    {
        output[i] = 0;
    }
}
```

```
volatile float *matrix1 = (float *)0x0070;
volatile float *matrix2 = (float *)0x0094;
volatile float *output = (float *)0x00B8;

void initialization(){
    for (uint8_t i = 0; i<9; i++)
    {
        matrix1[i] = i;
    }
    for (uint8_t i = 0; i<9; i++)
    {
        matrix2[i] = i + 0.5879;
    }
    for(uint8_t i=0;i<9;i++)
    {
        output[i] = 0;
    }
}
```

Η αρχικοποίηση των πινάκων γίνεται ως εξής:
Δημιουργούνται 3 pointers που δείχνουν στην θέση της μνήμης που δείχνει ο πίνακας. Στη συνέχεια με μια 1 for λούπα βάζουμε τιμές στις θέσεις του πίνακα ώστε να μπορούμε να κάνουμε τη πράξη και να ελέγξουμε την ορθότητα των αποτελεσμάτων. Ο Πίνακας του αποτελέσματος αρχικοποιήται στο 0.

Σημαντικό είναι να ελέγξουμε τις θέσεις που αρχικοποιούμε στη μνήμη ώστε να μην πανωγράφονται τα στοιχεία του πίνακα. Έτσι όταν οι πίνακες έχουν μέσα integers που είναι 2 bytes στην συγκεκριμένη αρχιτεκτονική δίνουμε διαφορετικές αρχικές θέσεις όπως εξηγούνται παρακάτω στο RAM MAP.

RAM MAP

Θέσεις μνήμης πινάκων INT:

Matrix1

0x70-71	0x72-73	0x74-75
Matrix1ArrayItem(0)	Matrix1ArrayItem (1)	Matrix1ArrayItem (2)

0x76-77	0x78-79	0x7A-7B
Matrix1ArrayItem (3)	Matrix1ArrayItem (4)	Matrix1ArrayItem (5)

0x7C-7D	0x7E-7F	0x80-81
Matrix1ArrayItem (6)	Matrix1ArrayItem (7)	Matrix1ArrayItem (8)

Matrix2

0x82-83	0x84-85	0x86-87
Matrix2ArrayItem(0)	Matrix2ArrayItem (1)	Matrix2ArrayItem (2)

0x88-89	0x8A-8B	0x8C-8D
Matrix2ArrayItem (3)	Matrix2ArrayItem (4)	Matrix2ArrayItem (5)

0x8E-8F	0x90-91	0x92-93
Matrix2ArrayItem (6)	Matrix2ArrayItem (7)	Matrix2ArrayItem (8)

Output

0x94-95	0x96-97	0x98-99
output(0)	output(1)	output(2)

0x9A-9B	0x9C-9D	0x9E-9F
output(3)	output(4)	output(5)

0x100-101	0x102-103	0x104-105
output(6)	output(7)	output(8)

Θέσεις μνήμης πινάκων FLOAT:

Matrix1

0x70-73	0x74-77	0x78-7B
Matrix1ArrayItem(0)	Matrix1ArrayItem (1)	Matrix1ArrayItem (2)

0x7C-7F	0x80-83	0x84-87
Matrix1ArrayItem (3)	Matrix1ArrayItem (4)	Matrix1ArrayItem (5)

0x88-8B	0x8C-8F	0x90-93
Matrix1ArrayItem (6)	Matrix1ArrayItem (7)	Matrix1ArrayItem (8)

Matrix2

0x94-97	0x98-9B	0x9C-9F
Matrix2ArrayItem(0)	Matrix2ArrayItem (1)	Matrix2ArrayItem (2)

0x100-103	0x104-107	0x108-10B
Matrix2ArrayItem (3)	Matrix2ArrayItem (4)	Matrix2ArrayItem (5)

0x10C-10F	0x110-113	0x114-117
Matrix2ArrayItem (6)	Matrix2ArrayItem (7)	Matrix2ArrayItem (8)

Output

0x118-11B	0x11C-11F	0x120-123
output(0)	output(1)	output(2)

0x124-127	0x128-12B	0x12C-12F
output(3)	output(4)	output(5)

0x130-133	0x134-137	0x138-13B
output(6)	output(7)	output(8)

Η συνάρτηση matrixmult

Η συνάρτηση αυτή είναι ίδια και στα 2 προγράμματα και κάνει τη πράξη του πολλαπλασιασμού πινάκων όπως τη γνωρίζουμε από προηγούμενα μαθήματα. Ουσιαστικά αποτελείται από 2 εμφολευμένες for.

```
void matrixmult(){
    for(uint8_t i=0; i<3; i++)
    {
        for(uint8_t j=0; j<3; j++)
        {

        output[3*i+j] = matrix1[3*i]*matrix2[j] + matrix1[3*i+1]*matrix2[j+3] + matrix1[3*i+2]*matrix2[j+6];

        }
    }
}
```

Χρήση πόρων του μικροελεγκτή στις 2 λύσεις

Παρατηρούμε ότι το πρόγραμμα που εκτελεί τον πολλαπλασιασμό με τους Float αριθμούς χρειάζεται πολλούς περισσότερους κύκλους ρολογιού για να εκτελέσει τις πράξεις. Μάλιστα η διαφορά που βλέπουμε στους συνολικούς κύκλους εκτέλεσης του πολλαπλασιασμού είναι σχεδόν 7 φορές μεγαλύτερος στο πρόγραμμα με τους Float.

Πιο συγκεκριμένα, ο πολλαπλασιασμός με Float αριθμούς χρειάζεται, 7264 κύκλους ρολογιού ενώ η ίδια πράξη για integers χρειάζεται 1100 κύκλους ρολογιού.

Αυτά τα αποτελέσματα οφείλονται σε πολλούς παράγοντες. Ένας από αυτούς είναι ότι οι integers είναι 2 bytes σε μέγεθος ενώ οι float είναι 4 bytes. Αυτό όμως δεν είναι και ο πιο σημαντικό λόγος. Βασικός λόγος που οι πράξεις παίρνουν περισσότερους κύκλους είναι επειδή στην αριθμητική με floats ο επεξεργαστής πρέπει να ορίσει σε ποιο σημείο βρίσκεται η υποδιαστολή σε κάθε αριθμό και να κάνει πολλαπλές πράξεις για να ορίσεις το σημείο της στο αποτέλεσμα. Έτσι παρατηρούμε ότι καθώς τρέχουμε τα προγράμματα και βλέποντας το κώδικα που παράγει το atmel studio ο κώδικα που παράγει για τον πολλαπλασιασμό με float είναι πολλες φορές περισσότερος σε σχέση με τον κώδικα που παράγει για integers.

Τελικό Αποτέλεσμα

Τα προγράμματα τρέχουν ξεκινώντας με αρχικοποίηση των πινάκων και στη συνέχεια με τη πράξη του πολλαπλασιασμού τους. Τελικά παράγεται ο πίνας output που έχεις τα αποτελέσματα των πράξεων. Παρατήρησα ότι οι πράξεις γίνονται σωστά και στα 2 προγράμματα χωρίς προβλήματα.