Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Σχεδιασμός Ενσωματωμένων Συστημάτων
Εργαστηριακή άσκηση 1

Ομάδα Α17
Μαρίνος Δημολιάνης 03111602
Κωνσταντίνος Τουλούπας 03111018

Optimization του αλγορίθμου phods για εκτίμηση κίνησης

Ζητούμενο 1ο

Στο πρώτο ερώτημα μετασχηματίζουμε τον δοθέντα κώδικα, ώστε να μετράται ο χρόνος που χρειάζεται η συνάρτηση phods_motion_estimation() για να εκτελεστεί. Η μέτρηση έγινε με τη χρήση της gettimeofday(), η οποία μας επιστρέφει μετά την εκτέλεση τον χρόνο που έχει περάσει από την 1/1/1970 σε δευτερόλεπτα και μικροδευτερόλεπτα.

Κώδικας για μέτρηση χρόνου

```
#define MILLION 1000000L
#include <time.h>
/*Υπόλοιπος κώδικας*/
int main(){
/*Υπόλοιπος κώδικας*/
struct timeval tts,ttf;
     double ttotal;
     read_sequence(current,previous);
     gettimeofday(&tts,NULL);
     phods_motion_estimation(current,previous,arr1,arr2);
     gettimeofday(&ttf,NULL);
     ttotal= MILLION* (ttf.tv_sec - tts.tv_sec)+ ttf.tv_usec-
tts.tv_usec;
            printf("Time elapsed: %f usec\n",ttotal);
            return 0;
}
```

Τα αποτελέσματα της τροποποίησης φαίνονται παρακάτω. Η εκτέλεση της phods_motion_estimation() διαρκεί περίπου 130000 μικροδευτερόλεπτα.

```
root@debian-armhf:~# gcc -O0 phods.c -o phods
root@debian-armhf:~# ./phods
Time elapsed: 130688.000000 usec
root@debian-armhf:~# ./phods
Time elapsed: 132364.000000 usec
```

Ζητούμενο 2ο

Στο δεύτερο ερώτημα της άσκησης μας ζητήθηκε ο μετασχηματισμός του δοθέντος κώδικα με τη χρήση αλγοριθμικών μετασχηματισμών βελτιστοποίησης. Επιχειρήσαμε να χρησιμοποιήσουμε όλους τους δυνατούς μετασχηματισμούς ωστόσο δεν απέδωσαν όλοι speedup. Ειδικότερα, οι τεχνικές που μας έδωσαν μείωση στο χρόνο εκτέλεσης είναι οι εξής:

- Loop unrolling
- Loop merging

Ενώ οι μετασχηματισμοί με τους οποίους δεν είδαμε κάποιο speedup είναι:

- Loop tilling
- Loop interchange

Πιο συγκεκριμένα, εντός των δυο βασικών loop αφαιρέσαμε την *while(S>0)* καθώς και την *for(i=-S;i<S+1;i+=S)* και εφαρμόσαμε loop unrolling. Παρατηρήθηκε ότι οι παραπάνω βρόχοι σε κάθε επανάληψη των εξωτερικών βρόχων x,y είχαν πανομοιότυπη συμπεριφορά και δεν είχαν καμία εξάρτηση από τα x,y. Η μεταβλητή S σε κάθε επανάληψη(των x και y) έπαιρνε τις τιμές {-4,0,4,-2,0,2,-1,0,1} , για αυτό εφαρμόσθηκε loop unrolling με αυτές τις τιμές τετριμμένες και όχι υπό την παρουσία loop.

Εν συνεχεία παρατηρήσαμε ότι θα μπορούσαν να ομαδοποιηθούν οι παραπάνω τιμές ανά τριάδα καθώς η while που κάναμε unroll περιελάμβανε σε κάθε επανάληψη της μια τριάδα του S(-S,0,S) . Εκμεταλλευόμενοι αυτή την ιδιότητα του προγράμματος συγχωνεύσαμε εντός των δύο loop(k,l) το σώμα της *for(i=-S;i<S+1;i+=S)* με αποτέλεσμα να συμπυκνωθούν τα loops και την μείωση του χρόνου εκτέλεσης του προγράμματος. Προφανώς λόγω του unroll και του merge απαιτήθηκαν επιπρόσθετες τροποποιήσεις στον κώδικα όπως αρχικοποιήσεις μεταβλητών καθώς και δημιουργία νέων για να επιτευχθούν τα παραπάνω optimization.

Επίσης εκτός των παραπάνω τεχνικών αντικαταστάθηκαν οι μεταβλητές των οποίων οι τιμές ήταν γνωστές και δεν χρειαζόταν κάθε φορά η φόρτωση τους, από τις ίδιες τις τιμές τους . Ακόμη οι πιο συχνά χρησιμοποιούμενες μεταβλητές των loops ανατέθηκαν σε registers ώστε να επιταχυνθεί η εκτέλεση του προγράμματος.

Ζητούμενο 3ο

Στο τρίτο ερώτημα ζητείται η παραμετροποίηση του κώδικα ώστε να μπορεί να εκτελεσθεί το πρόγραμμα για διάφορα μεγέθη τετραγωνικών blocks , δηλαδή για διάφορες τιμές του B . Με την εφαρμογή της τεχνικής Design Space Exporation θέλουμε να προκύψει το βέλτιστο μέγεθος τετραγωνικού Block πλευράς B για το οποίο εξασφαλίζεται ο μικρότερος χρόνος εκτέλεσης του κώδικα.

Για να επιτευχθεί αυτό κατασκευάσαμε ένα script το οποίο εκτελεί την optimized έκδοση της phods όπως αυτή προέκυψε στο ερώτημα για κάθε B που είναι κοινός διαιρέτης των διαστάσεων του πίνακα. Στο εν λόγω script εκτελούμε το πρόγραμμα 3 φορές για κάθε B και παίρνουμε τον μέσο όρο των τριών εκτελέσεων και το αποθηκεύουμε σε ένα αρχείο. Το B παίρνει τις τιμές 2,4,8 και 16. Τέλος ταξινομούμε τους χρόνος που προκύπτουν σε αύξουσα σειρά ώστε να διακρίνονται εύκολα οι μικρότεροι χρόνοι. Μετά από την εκτέλεση του script, τα αποτελέσματα αποθηκεύονται στο αρχείο Results3. Τις περισσότερες φορές καταλήξαμε ότι το ιδανικό μέγεθος για το B είναι 16. Το speedup που προκύπτει τελικά με block size 16 είναι κοντά στο 4(δηλαδή 35000 μικροδευτερόλεπτα ο χρόνος εκτέλεσης).

Κώδικας για την παραμετροποίηση

```
int main(int argc, char *argv[])
{

     int current[144][176], previous[144][176], i, j;
     if (argc != 2){
          /*Incorrect number of arguments*/
          usage(argv[0]);
     }
     B = atoi(argv[1]);
     if ( (  (B!= 2)&&(B != 4)&&(B != 8)&&(B != 16)   )) {
          /*If B is not a common divider of 144 and 176*/
          fprintf(stderr, "`%s' is not valid for block size\n",
argv[1]);
          exit(1);
     }
     int *arr1[144/B],*arr2[144/B] ;
     for(i=0;i<(144/B);i++)
     {
          arr1[i] = (int*)malloc((176/B)*sizeof(int));
          arr2[i] = (int*)malloc((176/B)*sizeof(int));
     }
     /*initialize to 0 */
     for(i = 0; i< (144/B); i++){
          for(j=0; j<(176/B); j++){
               arr1[i][j] = 0;
          }
     }
     /*Υπόλοιπη main*/
     for ( i = 0; i < 144/B; i++)
     {
          int* currentIntPtr = arr1[i];
          free(currentIntPtr);
          currentIntPtr = arr2[i];
          free(currentIntPtr);
     }
     //printf("Time elapsed: %f usec\n",ttotal);
     return 0;
}
```

Ζητούμενο 4ο

Στο τέταρτο μέρος της άσκησης καλούμαστε να εφαρμόσουμε Design Space Exploration αναφορικά με την εύρεση του βέλτιστου μεγέθους ορθογώνιου block διαστάσεων Bx , By.
Τροποποιώντας τον κώδικα ώστε να δέχεται δύο παραμέτρους και με την κατασκευή ενός script ίδιας φιλοσοφίας με αυτό του παραπάνω ερωτήματος αναζητήσαμε block τέτοιο ώστε το Bx να είναι διαιρέτης του N και το By να είναι διαιρέτης του M.

Μετά την εκτέλεση αρκετών φορών του script και ταξινομώντας τους χρόνους σε αύξουσα σειρά δεν παρατηρήθηκε κάποιος ακριβής συνδυασμός (Bx,By) για τον οποίο να προκύπτει πάντα ο ελάχιστος χρόνος. Ωστόσο θα μπορούσε να περιοριστεί η αναζήτηση σε μεγάλα By καθώς για μεγάλα By παρατηρούσαμε συνήθως τους μικρότερους χρόνους. Με

αυτόν τον τρόπο ίσως θα μπορούσαμε ευριστικά να περιορίσουμε τον χώρο λύσεων και να έχουμε ένα αρκετά ικανοποιητικό αποτέλεσμα χωρίς την αναζήτηση όλου του χώρου λύσεων. Ένας άλλος τρόπος θα μπορούσε να είναι ο περιορισμός της αναζήτησης για ορθογώνια πλαίσια με εμβαδόν μεγαλύτερο ή ίσο από μια συγκεκριμένη τιμή. Παρόλο που δεν προκύπτει σταθερό μέγεθος block για το οποίο να παρατηρείται η μέγιστη απόδοση του προγράμματος το speedup των καλύτερων δυνατών block είναι κοντά στο 5(25000-30000 μικροδευτερόλεπτα).Τα αποτελέσματα αποθηκεύονται στο αρχείο Results4, με αύξουσα σειρά κατά τον χρόνο εκτέλεσης της phods_motion_estimation().

Στο αρχείο .zip είναι οι κώδικες ert1.c έως ert4.c που αντιστοιχούν σε κάθε ερώτημα, καθώς και τα δύο scripts που χρησιμοποιήσαμε.