Λειτουργικά Συστήματα

2η Άσκηση

Αναγνωστόπουλος Αθανάσιος

AM: 1115201800006

Αρχικά υπάρχουν 2 αρχεία .c τα main.c και list.c και ένα αρχείο list.h στο οποίο ορίζονται οι συναρτήσεις που χρησιμεύουν στην για την δομή δεδομένων που χρησιμοποιώ (λίστα).

```
typedef struct {
    unsigned long long time;
    int page;
    bool dBit;                /* Used for writes counter */
    bool write;              /* write = 1 if operation is "W" , if oper
ation is "R" => write = 0 */
    bool full;               /* Indicates if the frame is full or not *
/
    bool refBit;            /* Used for SecondChance algorithm */
}frame;
```

Η Ram ορίζεται ως ένας πίνακας από frames μεγέθους RamSize (το οποίο δίνεται ως όρισμα). Το struct frame περιέχει τον χρόνο που θα γίνει εισαγωγή το page καθώς και το ίδιο το page. Ακόμα περιέχει μία μεταβλητή τύπου bool (dirty Bit) που χρησιμοποιείται για την αύξηση των writes, μια μεταβλητή (write) που δείχνει αν το operation που θα γίνει είναι read ή write, μια μεταβλητή (full) που δείχνει αν το frame είναι γεμάτο ή όχι και τέλος το reference Bit που χρησιμοποιείται στον αλγόριθμο Δεύτερης Ευκαιρίας (SecondChance).

```
typedef struct node{
   int frame;
   int page;
   struct node* next;
}hashBlock;
```

Το Hash Table είναι ένας πίνακας μεγέθους RamSize (δηλαδή όσα και τα πλαίσια (frames) της κεντρικής μνήμης) που περιέχει δείκτες σε λίστες. Κάθε node της λίστας (ή αλλιώς hashBlock) δείχνει σε ποιο frame έχει μπει ένα συγκεκριμένο page. Κάθε φορά που ένα page εισάγεται ή αφαιρείται στην Ram ανανεώνεται και το Hash Table.

Σε περίπτωση που το page υπάρχει ήδη στο Hash Table ανανεώνουμε τον χρόνο εισαγωγής του στη μνήμη καθώς και το reference bit σε 1 (“true”).

Σε περίπτωση που το Page δεν υπάρχει στο Hash Table έχουμε δύο περιπτώσεις.

α. Αν υπάρχει διαθέσιμος χώρος στην κεντρική μνήμη το εισάγουμε στην πρώτη άδεια θέση που υπάρχει καθώς και στο Hash Table.

β. Αν η Ram είναι γεμάτη τότε καλούμε τον αλγόριθμο αντικατάστασης (LRU ή SecondChance αναλόγως το όρισμα) που έχει δοθεί ώστε να μας επιστρέψει ποιο page θα αντικαταστήσουμε. Στην θέση του εισάγουμε το καινούργιο page και ανανεώνουμε τον χρόνο εισαγωγής. Επίσης αφαιρούμε το παλιό page από την Ram αλλά και από το Hash Table.

Σημείωση: Σε κάθε εκτέλεση του προγράμματος μπορεί να υπάρχει μία μικρή απόκλιση στα page faults, στα reads αλλά και στα writes. Αυτό ευθύνεται στο γεγονός ότι η συνάρτηση current_time_in_ms() που χρησιμοποιείται για την καταμέτρηση

του χρόνου δεν έχει τόσο μεγάλη ακρίβεια όσο χρειάζεται με αποτέλεσμα κάποια pages να εισάγονται στην Ram το ίδιο millisecond και έτσι να αλλάζουν τα pages που επιστρέφουν οι αλγόριθμοι LRU και SecondChance (εφόσον υπάρχουν και pages με τον ίδιο χρόνο εισαγωγής).

Μεταγλώττιση

gcc –o main main.c list.c -lm

Εκτέλεση

Το πρόγραμμα δέχεται 3 ορίσματα π.χ. ./main LRU 100 150

Όπου:

Το πρώτο όρισμα καθορίζει τον αλγόριθμο αντικατάστασης που θα χρησιμοποιηθεί (“LRU” ή “SecondChance”).

Το δεύτερο όρισμα καθορίζει τον αριθμό πλαισίων (frames) της κεντρικής μνήμης (Ram).

Τέλος το τρίτο όρισμα θα καθορίζει το πλήθος αναφορών q που διαβάζει το πρόγραμμα εναλλάξ από το κάθε αρχείο.

Επίσης με #define προσδιορίζεται ο μέγιστος αριθμός αναφορών που θα εξεταστούν από τα αρχεία ίχνους (MAX_PAGES).