Προηγμένοι Μικροεπεξεργαστές

Παρουσίαση Projects

Γενικά

- 2 Θέματα για 1 άτομο
- 5 Θέματα για 2 άτομα
- 4 Θέματα για 3 άτομα

Γενικά

- Προθεσμία:
  - Τέλος εξεταστικής (Hard Deadline)
- Τυπικός απαιτούμενος χρόνος:
  - 3-4 μέρες προετοιμασία
  - 7-10 μέρες δουλειά full time
- Παραδοτέα:
  - Κώδικας και αναλυτική αναφορά

Debugging

- Το debugging είναι επίσης κομμάτι της δουλειάς σας
- Θα σας βοηθήσουμε αφού:
  - Έχετε ψάξει το manual της Intel
  - Έχετε ψάξει για πληροφορίες στο δίκτυο
  - Έχετε ψάξει εξονυχιστικά για να βρείτε το πρόβλημα
  - Έχετε περιορίσει το που μπορεί να οφείλεται το πρόβλημα

Κώδικας

- Ο κώδικας πρέπει να είναι:
  - Οργανωμένος (συναρτήσεις, structs, macros, ξεχωριστές λειτουργικότητες σε ξεχωριστά αρχεία)
  - Καθαρογραμμένος (περιγραφή συναρτήσεων, σχόλια μέσα στον κώδικα, ονόματα που βοηθούν την κατανόηση, στοίχιση του κώδικα)
  - Βαθμολογήστε και για την ποιότητα του τελικού κώδικα

Project 1 – Scheduler

- Scheduler λειτουργικού συστήματος
- Αναπτύσεται πάνω στον κώδικα των ασκήσεων
- Θα δίνει την δυνατότητα για round-robin scheduler, δυναμική εκκίνηση/διακοπή tasks και για προσωρινό σταμάτημα ενός task για ένα χρονικό διάστημα

Project 1 – Scheduler

- Διεπαφή με το υπόλοιπο σύστημα
  - int exec (function address)
  - int kill (int pid)
  - void sched()
  - int sleep(int pid, int ticks)
  - int wakeup(int pid)
  - void update_wakeup_timers()

Project 1 - Scheduler

- 3 άτομα
- Προτινόμενος χωρισμός
  - Απο κοινού σχεδιασμός των κοινόχρηστων δομών και τυχόν βοηθητικών συναρτήσεων
  - 1 άτομο exec(), kill()
  - 1 άτομο sched(), sleep(), wakeup()
  - 1 άτομο στήσιμο επίδειξης, δοκιμαστικών tasks κτλ

Project 1 – Scheduler

- Μία struct task_info θα περιέχει όλες τις πληροφορίες του scheduler για ένα task. Ενδεικτικά:
  - TSS Selector
  - Process ID (pid)
  - Πεδία active, running, ready
  - TimeToWakeup

Project 1 – Scheduler

- Ο scheduler θα δεσμεύει έναν πίνακα από task_info structs, μέσα στον οποίο θα βρίσκονται οι πληροφορίες για όλα τα tasks
- Ο πίνακας θα έχει μέγεθος NR_TASKS
- Πάνω σε αυτόν τον πίνακα θα ενεργούν όλες οι συναρτήσεις του scheduler

Project 1 – Scheduler

- Διεπαφή int exec (function address)
  - Λειτουργικότητα:
    - Απενεργοποιεί τις διακοπές
    - Δεσμεύει και αρχικοποιεί ένα task_info struct, μία περιοχή στοίβας και ένα TSS ώστε το task να εκτελέσει την συνάρτηση της οποίας η διεύθυνση δώθηκε σαν όρισμα
    - Τα συσχετίζει μεταξύ τους

Project 1 – Scheduler

- Διεπαφή int exec (function address)
  - Λειτουργικότητα (συνέχεια):
    - Θέτει το πεδίο active του task_info
    - Αυξάνει έναν μετρητή διεργασιών και τον αναθέτει στο πεδίο pid
    - Ενεργοποιεί τις διακοπές
    - Επιστρέφει το pid αν όλα πήγαν καλά ή -1 σε περίπτωση αποτυχίας.

Project 1 – Scheduler

- Διεπαφή int kill (int pid)
  - Λειτουργικότητα:
    - Απενεργοποιεί τις διακοπές
    - Βρίσκει το task_info που αντιστοιχεί στο pid ή αν το pid είναι -1 βρίσκει το task_info της τρέχουσας διεργασίας
    - Απενεργοποιεί το πεδίο active του task_info
    - Αποδεσμεύει τις δομές δεδομένων που χρησιμοποιούσε το task
    - Ενεργοποιεί τις διακοπές
    - Επιστρέφει το pid σε περίπτωση επιτυχίας, -1 σε περίπτωση αποτυχίας

Project 1 – Scheduler

- Διεπαφή void sched ()
  - Λειτουργικότητα:
    - Καλείται από την ISR του timer
    - Επιλέγει με round-robin αλγόριθμο διεργασίες που είναι active και ready
    - Επαναφέρουμε το πεδίο running της διεργασίας που διακόπηκε
    - Η επιλεγμένη διεργασία σημειώνεται σαν running και κρατάμε το pid της
    - Εκτελούμε task switching στην επιλεγμένη διεργασία

Project 1 – Scheduler

- Διεπαφή int sleep(int pid, int ticks)
  - Λειτουργικότητα:
    - Απενεργοποιεί τα interrupts
    - Βρίσκει το task_info που αντιστοιχεί στο pid
    - Επαναφέρει το πεδίο ready
    - Θέτει το πεδίο TimeToWakeup ίσο με ticks
    - Αν κάναμε sleep στο τρέχον task, καλούμε την sched() για να εκτελέσει κάποιο καινούργιο task
    - Ενεργοποιεί τα interrupts και επιστρέφει το pid ή -1 σε περίπτωση αποτυχίας

Project 1 – Scheduler

- Διεπαφή int wakeup(int pid)
  - Λειτουργικότητα:
    - Απενεργοποιεί τα interrupts
    - Βρίσκει το task_info που αντιστοιχεί στο pid
    - Θέτει το πεδίο ready
    - Ενεργοποιεί τα interrupts και επιστρέφει το pid ή -1 για αποτυχία

Project 1 – Scheduler

- Διεπαφή void update_wakeup_timers()
  - Λειτουργικότητα:
    - Καλείται από την timer ISR πριν τον sched
    - Βρίσκει τα task_info που δεν είναι ready και μειώνει το πεδιό TimeToWakeup
    - Αν κάποιο μηδενιστεί, θέτει το πεδίο ready του task_info

Project 1 - Scheduler

- Επιπλέον για να γίνει επίδειξη του συστήματος, θέλουμε:
  - 1 δοκιμαστικό tasks που απλώς να γράφει στην οθόνη, με κάποιον αλγόριθμο
    - Αυτό το task να μην χρησιμοποιεί global μεταβλητές
  - 1 idle task που θα περιέχει μόνο ένα idle loop και θα εκτελείται όταν δεν υπάρχει άλλο active και ready task
  - 1 task που θα χρησιμοποιεί την διεπαφή του scheduler για να ενεργοποιεί/απενεργοποιεί αντίγραφα του δοκιμαστικού task ή να τα κοιμίζει ώστε να γίνει εμφανής η λειτουργικότητα που προσφέρει ο scheduler

Project 2 – Memory Management

- Διαχείρηση μνήμης
- Πολύ βασική υλοποίηση των συναρτήσεων malloc και free της standard library της C
  - void *malloc(int size): δεσμεύει και επιστρέφει περιοχή μνήμης μεγέθους size
  - int free (void *mem_ptr): αποδεσμεύει την περιοχή μνήμης που δείχνει ο pointer

Project 2 – Memory Management

- Το συνολικότερο project θα στηθεί πάνω στον κώδικα των ασκήσεων
- Επιπλέον θα παραδωθεί:
  - Ένα αρχείο που θα υλοποιεί αυτές τις δύο συναρτήσεις
  - Δοκιμαστικός κώδικας που θα φορτώνει και θα ξεφορτώνει αρχεία σε δυναμικά δεσμευμένη μνήμη, εκτύπωση σχετικών μηνυμάτων και κατάστασης μνήμης

Project 2 – Memory Management

- 2 άτομα
- Προτινόμενος χωρισμός:
  - 1 άτομο για την υλοποίηση της malloc και της free
  - 1 άτομο για τον κώδικα επίδειξης
  - Αλλά ίσως βολεύει να δουλέψετε και μαζί

Project 2 – Memory Management

- Λογική σχεδίασης:
  - Ο memory manager θα δεσμεύει 1MB για δυναμική μνήμη μετά το τέλος του αρχείου
  - Όλη η μνήμη μετά την τελευταία γραμμή του προγράμματος είναι διαθέσιμη

Project 2 – Memory Management

- Λογική σχεδίασης(συνέχεια):
  - Αυτό το 1MB, θα χωριστεί σε 256 κομμάτια των 4KB
  - Ένας πίνακας, θα κρατάει πληροφορίες ανάθεσης για καθένα από αυτά τα κομμάτια

Project 2 – Memory Management

- void *malloc(int size)
  - Η malloc θα ψάχνει στον προηγούμενο πίνακα για να βρει συνεχόμενα αδέσμευτα κομμάτια που να καλύπτουν το ζητούμενο size
  - Αν τα βρει
    - Σημειώνει τα κομμάτια σαν δεσμευμένα με έναν μοναδικό αύξων αριθμό ανάθεσης
    - Επιστρέφει την διεύθυνση όπου ξεκινά το πρώτο κομμάτι
  - Αν όχι επιστρέφει 0

Project 2 – Memory Management

- int free(void *mem_ptr)
  - Η free παίρνει σαν όρισμα την αρχή της περιοχής μνήμης που πρέπει να αποδεσμευτεί
  - Ξεκινά από το αντίστοιχο κομμάτι και σημειώνει τα διαδοχικά κομμάτια σαν αδέσμευτα μέχρι να τελειώσει η δεσμευμένη περιοχή

Project 3 – Stdio

- Υλοποίηση βασικών συναρτήσεων εκτύπωσης στην οθόνη και διαβάσματος από το πληκτρολόγιο, μέσω DOS services
  - getchar, putchar
  - gets, puts
  - scanf, printf

Project 3 – Stdio

- 3 άτομα
  - 1 άτομο για την υλοποίηση των βασικών συναρτήσεων
  - 1 άτομο για την υλοποίηση βοηθητικών συναρτήσεων που θα μετατρέπουν μεταβλητές σε strings και ανάποδα
    - Απεικόνιση σαν decimal, binary, hex
  - 1 άτομο για την ύλοποιηση των printf, scanf που θα χρησιμοποιούν υπόλοιπες συναρτήσεις

Project 4 - Keyboard

- Υλοποίηση οδηγών σε protected mode για το πληκτρολόγιο (PC/AT)
  - Χαμηλού επιπέδου προσπέλαση στο πληκτρολόγιο με χρήση interrupts
  - Συνάρτηση που να παρέχει την λειτουργικότητα του int 0x21, ah = 0x07

Project 4 - Keyboard

- 3 άτομα
  - Επέκταση της άσκησης του editor, ώστε να μην χρειάζεται το task RM για το διάβασμα του πληκτρολογίου
  - Οι οδηγοί θα χρησιμοποιούν το IRQ1 (int 9) και τα I/O ports του πληκτρολογίου
  - Θα επιστρέφουν τον χαρακτήρα που πατήθηκε, αναλύοντας τα scan codes του πληκτρολογίου

Project 4 - Keyboard

- Επιπλέον, θα υλοποιηθεί ένα υποκατάστατο του int 0x21, 0x07 έτσι ώστε να παρέχει την ίδια λειτουργικότητα (άμεσα σε protected mode)
- Η συνάρτηση θα πρέπει να μην προκαλεί busy-waiting, αλλά να δίνει τον έλεγχο στον scheduler, μέχρι να χτυπήσει το IRQ1

Project 4 - Keyboard

- Για περισσότερες πληροφορίες:
  - Περιγραφή του hardware interface του πληκτρολογίου στο Art of Assembly κεφάλαιο 20

Project 5 - Editor

- Επέκταση του editor των ασκήσεων, δίνοντας του πιο vi-like δυνατότητες
  - Υλοποίηση διακριτών normal, insert και command modes
  - Βελτίωση του display κομματιού

Project 5 - Editor

- 3 άτομα
  - 2 άτομα για τον parser
  - 1 άτομο για την απεικόνιση

Project 5 - Editor

- Το normal mode θα προσφέρει:
  - Είσοδο σε insert mode με 'i'
  - Είσοδο σε insert mode και σε καινούργια γραμμή με 'o'
  - Είσοδο σε command mode με ':'
  - Διαγραφή τρέχοντος χαρακτήρα με 'x'
  - Διαγραφή τρέχουσας γραμμής με 'dd'

Project 5 - Editor

- Το command mode θα προσφέρει:
  - Ακύρωση και επιστροφή σε normal mode με 'Escape'
  - Άνοιγμα αρχείου με ':e <file>'
  - Αποθήκευση αρχείου με ':w <file>'
  - Έξοδος με ':q'
  - Μετακίνηση στην γραμμή <line> με ':<line>'

Project 5 - Editor

- Το insert mode θα προσφέρει:
  - Εισαγωγή κειμένου
  - Επιστροφή σε normal mode με 'Escape'

Project 5 - Editor

- Ο κώδικας της απεικόνισης:
  - θα γίνει πιο καθαρός και δομημένος
  - Θα δείχνει το κείμενο που παράγεται από το command mode
  - Θα επιτρέπει scrolling

Project 6 – BMP Displayer in C

- Υλοποίηση ενός προγράμματος που θα διαβάζει ένα bitmap αρχείο και θα το απεικονίζει στην οθόνη

Project 6 – BMP Displayer in C

- 1 άτομο
  - Χρήση της VGA σε graphics mode
    - 640x480 16 colors
  - Φόρτωση μέσω DOS services της εικόνας από αρχείο στην μνήμη
  - Μετατροπή της εικόνας στο format της VGA
  - Εγγραφή στην video memory
    - Graphics mode: 0xA0000 (64 KB)

Project 6 – BMP Displayer in C

- Το πρόγραμμα θα είναι γραμμένο στο μεγαλύτερο κομμάτι του σε C,
  - αλλά θα εκτελείται μέσα από protected mode assembly
  - θα καλεί με την σειρά του assembly κώδικα για να χειριστεί τα BIOS και DOS services
  - Sample κώδικας για το πως γίνεται αυτό καθώς και scripts δίνονται έτοιμα

Project 7 – Sound Generation

- Υλοποίηση οδηγών για το PC Speaker και υλοποίηση προγράμματος που να αναπαράγει ένα μικρό κομμάτι μουσικής με χρήση αυτών των οδηγών

Project 7 – Sound Generation

- 1 άτομο
- Πληροφορίες για το PC Speaker:
  - http://fly.cc.fer.hr/GDM/articles/sndmus/speaker1.html

Project 8 – Compiler Flags

- Χρήση Performance Counters για την μελέτη της συμπεριφοράς μίας εφαρμογής όταν γίνεται compile με διάφορα optimization flags

Project 8 – Compiler Flags

- 2 άτομα
- Οι compilers
  - επιτρέπουν την ενεργοποίηση διαφόρων βελτιστοποιήσεων
  - Παρέχουν σημαίες που ενεργοποιούν την παραγωγή κώδικα για συγκεκριμένες γενιές επεξεργαστών
- Σκοπός:
  - ο έλεγχος του πως επηρεάζουν τα πιο συνηθισμένα optimization flags την απόδοση ενός προγράμματος

Project 9 - Debugger

- Υλοποίηση ενός απλού debugger χρησιμοποιώντας τα debug facilities των x86.

Project 9 - Debugger

- 2 άτομα
- Ο debugger πρέπει να έχει τις εξής interactive δυνατότητες:
  - Εκκίνηση μέσω του debugger μίας διεργασίας
  - Είσοδος στο debug περιβάλλον με τη σύλληψη του breakpoint interrupt
  - Δυνατότητα για εκτύπωση των περιεχομένων όλων των registers
  - Δυνατότητα για εκτύπωση της τιμής οποιασδήποτε θέσης μνήμης

Project 10 – Java Virtual Machine

- Διερεύνηση με performance counters της συμπεριφοράς διαφόρων java virtual machines

Project 10 – Java Virtual Machine

- 2 άτομα
- Java virtual machine
  - μετάφραση των java bytecodes
  - στήσιμο του java περιβάλλοντος
- Εκτός από την υλοποίηση της Sun (Hot Spot JVM) υπάρχει μία σειρά από άλλες virtual machines
  - Jikes, Kaffe, JamVM, CACAO

Project 10 – Java Virtual Machine

- διαφορετικές λογικές λειτουργίας
- πειραματικές ιδιότητες και βελτιστοποιήσεις
- έμφαση σε διαφορετικά είδη προγραμμάτων χρήστη

Project 11 - Paging

- Υλοποίηση paging μηχανισμού πάνω στον κώδικα της άσκηση 5

Project 11 - Paging

- 2 άτομα
  - Στο initialization του protected mode θα προστεθεί κώδικας που θα στήνει και θα ενεργοποιεί το paging (Page Directories, Page Tables)
  - Θα προστεθεί ένα task το οποίο θα έλεγχει το paging
  - Κάθε φορά που θα καλείται, θα κάνει rellocate τον κώδικα με τέτοιο τρόπο ώστε να μην διαταράσσεται η εκτέλεση του αλγόριθμου