

Σειρά Ασκήσεων 12: Μονάδες Εισόδου/Εξόδου (I/O), Σύνδεσμοι/Αρτηρίες, DMA

Προθεσμία: έως Πα. 7 Ιουνίου 2013, ώρα 11:20 π.μ., on-line ή σε χαρτί

Βιβλίο (Ελληνικό, έκδοση 4) - διαβάστε τα εξής, και με τους εξής τρόπους:

- DRAM, Διαφύλλωση (Interleaving) Μνήμης - σελίδες 546-551 (από §5.2) (καθώς και ενότητα 12.0 εδώ παρακάτω): διαβάστε τις κανονικά, αποτελούν μέρος της εξεταστέας ύλης.
- Αρτηρίες (Δίαυλοι - Buses), και Σύνδεσμοι (point-to-point high-speed-serial links) - σελίδες 678-681 (πρώτο μέρος της § 6.5): διαβάστε τις κανονικά, αποτελούν μέρος της εξεταστέας ύλης.
- Επικοινωνία Επεξεργαστή, Συσκευών I/O, και Μνήμης (DMA) - σελίδες 684-696 (§ 6.6), καθώς και οι σημειώσεις και ασκήσεις εδώ παρακάτω: διαβάστε τις κανονικά, αποτελούν μέρος της εξεταστέας ύλης.
- Οργάνωση I/O επεξεργαστών x86 - σελίδες 682-683 (από § 6.5): διαβάστε τις "στα γρήγορα" - εγκυκλοπαιδικά.
- Μαγνητικοί (σκληροί) δίσκοι και μνήμες flash - σελίδες 670-678 (§ 6.3 και 6.4): διαβάστε τις "στα γρήγορα" - εγκυκλοπαιδικά.
- Συστήματα Αποθήκευσης (Storage), RAID, φερεγγυότητα, αξιοπιστία, διαθεσιμότητα - σελίδες 668-670 και 700-708 (§ 6.2 και 6.9): σημαντικό θέμα για περαιτέρω προσωπική σας μελέτη, αλλά εκτός ύλης αυτού του μαθήματος.
- Παράδειγμα server για data centers, και άλλα εγκυκλοπαιδικά θέματα - σελίδες 708-723 (§ 6.10 και 6.12): χρήσιμα για εγκυκλοπαιδική ενημέρωση, αλλά εκτός ύλης αυτού του μαθήματος.

12.0 SRAM, DRAM, Προσπελάσεις Συνεχόμενων Λέξεων, Διαφύλλωση (Interleaving):

SRAM-DRAM: Όπως είπαμε στο μάθημα, τα chips μνήμης είναι οργανωμένα εσωτερικά σε κάμποσα blocks από στοιχεία μνήμης. Στις "στατικές" μνήμες (SRAM - Static Random Access Memory), τα στοιχεία μνήμης είναι flip-flops (με 6 transistors καθένα), και η αποθηκευμένη πληροφορία διατηρείται όσο είναι αναμένη η τάση τροφοδοσίας. Στις "δυναμικές" μνήμες (DRAM - Dynamic Random Access Memory), τα στοιχεία μνήμης είναι πυκνωτές (capacitors --ένας πυκνωτής και ένα transistor ανά bit), όπου αποθηκεύεται δυναμικά η πληροφορία. Λόγω του ρεύματος διαρροής, η πληροφορία αυτή (φορτίο στον πυκνωτή), χάνεται μέσα σε λίγα χιλιοστά του δευτερολέπτου (ms). Για να διατηρηθούν τα περιεχόμενα της DRAM πρέπει να τα αναζωογονούμε (refresh), δηλαδή να τα διαβάζουμε και να τα ξαναγράφουμε, κάθε περίπου 8 με 16 ms.

Μέγεθος (Χωρητικότητα - Mbits): Παρά το μειονέκτημά τους αυτό, και παρά την μεγαλύτερη καθυστέρηση προσπέλασης που έχουν, οι DRAM διαθέτουν ένα σημαντικό πλεονέκτημα: προσφέρουν περίπου μία τάξη μεγέθους μεγαλύτερη χωρητικότητα (capacity, Mbits --όχι "capacitance") ανά chip σε σχέση με τις SRAM. Έτσι, οι DRAM χρησιμοποιούνται σχεδόν πάντα για την κατασκευή της κύριας μνήμης (main memory) των υπολογιστών, ενώ οι SRAM χρησιμοποιούνται σχεδόν πάντα για τις κρυφές μνήμες (cache memories), λόγω της χαμηλότερης καθυστέρησής τους. Με την πρόοδο της τεχνολογίας κατασκευής ολοκληρωμένων κυκλωμάτων (chips), η χωρητικότητα των

chips μνήμης συνεχώς αυξάνει. Τις τελευταίες δεκαετίες, ο ρυθμός αυτής της αύξησης ήταν: **τετραπλασιασμός (x4) χωρητικότητας κάθε τρία (3) χρόνια**. Σήμερα είναι λίγο χαμηλότερος αυτός ο ρυθμός αύξησης, και είναι πολύ πιθανό να συνεχίσει παρόμοια, αν και κάποτε μάλλον θα πέσει κι άλλο. Το 2013, η τεχνολογία των DRAM βρίσκεται γύρω στα 2 έως 16 Gbits ανά chip (π.χ. βλ. www.micron.com). Εμπορικά, την μνήμη των υπολογιστών τη βρίσκει κανείς σε μικρές πλακέτες (modules - DIMM), που η καθεμιά έχει πάνω της συνήθως 8 (ή 9) ή 16 (ή 18) chips. Έτσι, π.χ., ένα module με 8 chips των 256M x 8 bits = 2 Gbits καθένα είχε συνολική χωρητικότητα 16 Gbits = 2 GBytes. Όταν τα chips είναι 9 αντί 8, ή 18 αντί 16, τα επιπλέον chips χρησιμοποιούνται για αποθήκευση κωδίκων ανίχνευσης και διόρθωσης σφαλμάτων (ECC - error correction codes).

Γραμμές και Στήλες: Μέσα στο chip της μνήμης, το κάθε block είναι ένας περίπου τετράγωνος πίνακας από στοιχεία μνήμης, με γύρω στις 1024 ή παραπάνω γραμμές επί 1024 ή παραπάνω στήλες. Για να διαβάσουμε ένα στοιχείο μνήμης επιλέγουμε πρώτα τη γραμμή στην οποία ανήκει αυτό, δίνοντας τη διεύθυνση γραμμής (row address) στον αποκωδικοποιητή γραμμής, ο οποίος ανάβει ένα σύρμα (word line) που διατρέχει και ενεργοποιεί την επιθυμητή γραμμή. Όταν ανάψει το σύρμα αυτό, όλα τα στοιχεία μνήμης (bits) πάνω στη γραμμή αυτή διαβάζονται, δηλαδή τοποθετούν το καθένα την τιμή του (περιεχόμενο του) στο αντίστοιχο σύρμα στήλης (bit line) που διατρέχει τη στήλη του. Έτσι, στο κάτω μέρος του block της μνήμης, στις απολήξεις των συρμάτων στήλης, εμφανίζεται το περιεχόμενο όλων των bits που είναι αποθηκευμένα στην επιλεγείσα γραμμή. Ένας μεγάλος πολυπλέκτης επιλέγει τότε το bit που εμείς θέλαμε, βάσει της διεύθυνσης στήλης (column address), και το δίνει προς τα έξω. Η όλη αυτή διαδικασία, από την είσοδο της διεύθυνσης γραμμής μέχρι να βγει το τελικό bit στην έξοδο, διαρκεί αρκετό χρόνο (γύρω στα 30 ns για τις σημερινές DRAM, όταν μιλάμε από τα pins διεύθυνσης μέχρι τα pins δεδομένων του chip της DRAM, ενώ αυτή αυξάνει σημαντικά όταν προστεθεί και η καθυστέρηση από το chip του επεξεργαστή μέχρι το chip της DRAM και πίσω).

Γειτονικές Προσπελάσεις (sequential Accesses): Εάν μετά την παραπάνω διαδικασία, όμως, θέλουμε να διαβάσουμε και μερικά από τα "διπλανά" bits αυτού που μόλις διαβάσαμε, τότε αυτό μπορεί να γίνει πολύ γρηγορότερα: τα bits αυτά είναι "έτοιμα", στις απολήξεις των συρμάτων στήλης, και το chip της μνήμης μπορεί να τα αποστείλει στον αιτούντα την ανάγνωση (π.χ. τον επεξεργαστή) πολύ γρήγορα το ένα μετά το άλλο (περίπου 1 bit κάθε μισό με 1 ns σε καθένα από τα σύρματα δεδομένων (data) για τις σημερινές DRAM). Εκμεταλλευόμενοι τη δυνατότητα αυτή, πετυχαίνουμε να προσπελάσουμε μεγάλες ομάδες γειτονικών λέξεων (π.χ. cache lines (blocks)) με πολύ μικρή επιπλέον επιβάρυνση σε σχέση με την αρχική καθυστέρηση προσπέλασης της πρώτης λέξης της ομάδας.

Πρόκειται για παράδειγμα μιάς γενικότερης περίπτωσης: πολλές φορές, υπάρχει ένα σημαντικό κόστος (start-up overhead) εκκίνησης μιάς εργασίας (π.χ. μεταφράσας δεδομένων) που είναι (σχεδόν) ανεξάρτητο του μεγέθους της εργασίας (π.χ. του όγκου των δεδομένων), και στη συνέχεια ένα μικρότερο (ή πολύ μικρότερο) επιπλέον κόστος (incremental cost) για την κάθε επιπλέον μονάδα εργασίας (π.χ. για κάθε επιπλέον Byte ή λέξη δεδομένων που μεταφέρονται). Άλλα τέτοια παραδείγματα συναντάμε στους δίσκους, στις αρτηρίες (buses), στα δίκτυα, κλπ.

Διαφύλλωση (Interleaving): Η άλλη τεχνική που χρησιμοποιείται για την αύξηση της παροχής (throughput) μιάς μνήμης --κυρίως για προσπελάσεις σε τυχαίες διευθύνσεις και όχι τόσο για συνεχόμενες διευθύνσεις-- είναι η *Διαφύλλωση Μνήμης (Memory Interleaving)*. Με την τεχνική αυτή, όταν οι διευθύνσεις που αποστέλονται σ' ένα σύστημα μνήμης (π.χ. ένα chip) αναφέρονται σε διαφορετικά από τα blocks (memory banks) που αυτό περιέχει, τότε ο ρυθμός αποστολής τέτοιων διευθύνσεων --και ο αντίστοιχος ρυθμός έναρξης προσπελάσεων-- είναι πολύ ψηλότερος από τον ρυθμό προσπελάσεων σε κάθε ένα block (bank), δεδομένου ότι τα blocks (banks) δουλεύουν εν παραλλήλω, μ' ένα τρόπο που θυμίζει ομοχειρία (pipelining).

Άσκηση 12.1: Απεικόνιση Μνήμης των Μονάδων E/E (Memory Mapped I/O)

Όπως είπαμε στο μάθημα, ένας συνηθισμένος τρόπος επικοινωνίας επεξεργαστή-μονάδων εισόδου/εξόδου (E/E - περιφερειακές συσκευές) είναι η "απεικόνιση μνήμης" των μονάδων E/E (memory-mapped I/O). Σε τέτοια συστήματα, ένα μέρος του "χώρου" φυσικών διευθύνσεων αντιστοιχεί στην κύρια μνήμη του υπολογιστή, ενώ οι υπόλοιπες φυσικές διευθύνσεις αντιστοιχούν στις περιφερειακές συσκευές. Αυτό σημαίνει ότι εντολές load και store των οποίων η εικονική διεύθυνση μεταφράζεται σε τέτοιες "άλλες" φυσικές διευθύνσεις προκαλούν μεταφορά δεδομένων από την εκάστοτε επιλεγόμενη περιφερειακή συσκευή προς τον επεξεργαστή (load) ή αντίστροφα (store), αντί να διαβάζουν ή να γράφουν μία θέση κύριας μνήμης. Η προστασία των περιφερειακών συσκευών (π.χ. αρχεία στο δίσκο) από ανεπίτρεπτες/κακόβουλες, λανθασμένες, ή ταυτόχρονες προσβάσεις εξοφάλλίζεται με το να μην απεικονίζει το λειτουργικό σύστημα καμία εικονική σελίδα χρήστη σε φυσική σελίδα που αντιστοιχεί σε περιφερειακές συσκευές, και να "εμφανίζει" αυτές τις φυσικές σελίδες μόνο στο χώρο εικονικών διευθύνσεων του λειτουργικού συστήματος (πλήν περιπτώσεων ειδικών συσκευών και ειδικών χρηστών που επιτρέπεται να αποκτούν κατ'ευθείαν πρόσβαση σε αυτές).

Σαν απλοϊκό παράδειγμα, για τους σκοπούς αυτής της άσκησης, θεωρήστε ότι μιλάμε για ένα σύστημα κύριας μνήμης και συσκευών E/E που βλέπει φυσικές διευθύνσεις λέξεων (όχι bytes, δηλαδή έχουν ήδη αφαιρεθεί τα 2-3 LS bits της διεύθυνσης του επεξεργαστή) μεγέθους (οι φυσικές διευθύνσεις λέξεων) 11 bits. Τον αντίστοιχο χώρο φυσικών διευθύνσεων, μεγέθους 2048 λέξεων, αποφασίζουμε να μοιράσουμε ως εξής:

- 0xxxxxxx: 1024 λέξεις κύριας μνήμης (main memory).
- 10xxxxxxx: 512 λέξεις για μία "μεγάλη" περιφερειακή συσκευή (δηλαδή μία συσκευή που περιέχει έναν μεγάλο buffer δεδομένων E/E που θέλουμε να μπορεί να βλέπει ο επεξεργαστής).
- 11000011xx: 4 λέξεις κατανεμημένες σε 2 "μικρές" συσκευές E/E, την "IN" στις 2 χαμηλότερες διευθύνσεις, και την "OUT" στις 2 ψηλότερες, όπου η κάθε μία από αυτές τις συσκευές έχει 2 εσωτερικούς καταχωρητές --τον "status" στην ζυγή διεύθυνση, και τον "data" στη μονή διεύθυνση.

(α) Σχεδιάστε, χρησιμοποιώντας πύλες AND (οσωνδήποτε εισόδων) και NOT, τον αποκωδικοποιητή διευθύνσεων που επιλέγει τη συσκευή (και τον καταχωρητή) που πρέπει να ενεργοποιηθεί κάθε φορά. Είσοδος του αποκωδικοποιητή είναι τα 11 σύρματα φυσικής διεύθυνσης λέξεων από τον επεξεργαστή (μετά τη μετάφραση της εικονικής διεύθυνσης από το TLB). Ο αποκωδικοποιητής έχει 6 σύρματα εξόδου: 1 για την κύρια μνήμη, 1 για την μεγάλη συσκευή E/E, 2 για την IN, και 2 για την OUT. Για κάθε μία από τις IN και OUT, το πρώτο σύρμα λέει πότε απευθυνόμαστε στον καταχωρητή της "status" και το δεύτερο λέει πότε απευθυνόμαστε στον "data". Σχεδιάστε το κύκλωμα που γεννά αυτά τα 6 σύρματα, δείχνοντας προσεκτικά ποια σύρματα εισόδου και ποιās πολικότητας χρησιμοποιείτε σε κάθε πύλη AND.

(β) Σε ποιά λέξη μνήμης (στο δεκαδικό, αρχίζοντας από την 0), ή σε ποιά θέση του buffer της μεγάλης συσκευής (πάλι στο δεκαδικό, αρχίζοντας από την 0), ή σε ποιόν καταχωρητή ποιās συσκευής αναφέρεται κάθε μία από τις εξής φυσικές διευθύνσεις λέξεων που δίδονται στο δεκαεξαδικό σύστημα, ή ποιές από αυτές είναι παράνομες (ανύπαρκτες): 000, 00E, 100, 200, 2FF, 300, 3FF, 400, 4FF, 500, 5FF, 600, 608, 60A, 60E, 60F, 610, 680, 7FF, 80E.

Άσκηση 12.2: Καταχωρητές Κατάστασης, Busy Wait, Polling

Φυσικά, οι μονάδες E/E δεν είναι πραγματική μνήμη: συχνά, διαβάζοντας από ορισμένη διεύθυνση, δεν παίρνει ο επεξεργαστής την ίδια τιμή με αυτήν που είχε γράψει σε αυτή τη διεύθυνση την τελευταία φορά που έγραψε εκεί (ο επεξεργαστής) --παίρνει την τιμή που θέλει να του δώσει κάθε φορά η μονάδα E/E, η οποία τιμή συχνά αλλάζει με το χρόνο. Επίσης, τέτοιες αναγνώσεις από περιφερειακές συσκευές μπορούν να έχουν

"παρενέργειες" (side-effects), όπως π.χ. να θέτουν ή να μηδενίζουν σημαίες (flag bits) που υποδεικνύουν π.χ. ότι διαβάστηκε η παρούσα τιμή εισόδου και δεν έχει έλθει ακόμα η επόμενη (νέα) τιμή εισόδου. Ομοίως, εγγραφή σε ορισμένη διεύθυνση περιφερειακής συσκευής μπορεί να προκαλεί π.χ. μετάδοση της πληροφορίας σε κάποιο σύρμα/δίκτυο, και όχι πραγματική εγγραφή σε κάποια flip-flops που να μπορούμε αργότερα να τα διαβάσουμε, και ενδέχεται επίσης η εγγραφή αυτή να προκαλεί και άλλες παρενέργειες όπως π.χ. μηδενισμό ενός flag που υποδεικνύει ότι παρελήφθη η παρούσα τιμή εξόδου και ότι η συσκευή δεν είναι ακόμα έτοιμη να παραλάβει την επόμενη τιμή.

Επειδή οι μονάδες E/E δεν συμπεριφέρονται σαν πραγματική μνήμη, οι τιμές που διαβάζουμε ή γράφουμε στις διευθύνσεις τους πρέπει να μην κρατιούνται στην κρυφή μνήμη, ειδάλως θα διαβάζουμε παλιές τιμές ή αυτά που γράφουμε δεν θα φτάνουν όλα ή αμέσως στις συσκευές E/E. Αυτό, το να παρακάμπουν δηλαδή οι προσπελάσεις αυτές την κρυφή μνήμη, επιτυγχάνεται συνήθως με ένα επιπλέον bit στον πίνακα σελίδων, που να υποδεικνύει "non-cacheable pages", ή με το να αναγνωρίζει η κρυφή μνήμη την ειδική μορφή των φυσικών διευθύνσεων των συσκευών E/E.

Ένα άλλο σύστημα επικοινωνίας επεξεργαστή-συσκευών E/E, διαφορετικό από την απεικόνιση μνήμης των μονάδων E/E, είναι η ύπαρξη ειδικών εντολών εισόδου/εξόδου (I/O instructions) στο ρεπερτόριο εντολών του επεξεργαστή. Οι εντολές εισόδου μοιάζουν με τις load και οι εντολές εξόδου μοιάζουν με τις store, όμως οι εντολές E/E είναι "προνομούχες" (privileged), δηλαδή επιτρέπεται να εκτελούνται μόνο σε "kernel mode", και οι εντολές E/E ειδοποιούν την κρυφή μνήμη να μην παρέμβει. Κατά τα άλλα, στις αρτηρίες E/E, οι εντολές E/E μάλλον καταλήγει να δίνουν διευθύνσεις εντελώς ανάλογες προς αυτές που δίνουν οι εντολές load/store στα συστήματα με απεικόνιση μνήμης των μονάδων E/E.

Σε αυτή την άσκηση, θεωρήστε ότι η "μικρή" συσκευή IN της άσκησης 12.1 είναι μία συσκευή εισόδου από πληκτρολόγιο, με τους καταχωρητές "κατάστασης" (status) και "δεδομένων" (data) που είπαμε. Μόλις έλθει νέος χαρακτήρας από το πληκτρολόγιο, η συσκευή θέτει τον καταχωρητή κατάστασης στην τιμή 1, και θέτει τον καταχωρητή δεδομένων στην τιμή που αποτελεί τον κώδικα ASCII του χαρακτήρα που ήλθε. Ανάγνωση (από πλευράς επεξεργαστή) του καταχωρητή κατάστασης δεν έχει παρενέργειες, ενώ ανάγνωση του καταχωρητή δεδομένων προκαλεί μηδενισμό του καταχωρητή κατάστασης (μέχρι να έλθει ο επόμενος χαρακτήρας --έτσι ξεχωρίζουμε, αν πατηθεί το ίδιο πλήκτρο πολλές φορές, πόσες φορές πατήθηκε).

(α) Γράψτε μία διαδικασία (procedure) "**read_kbd_busywait_char()**" σε C (ή, στην ανάγκη, σε ψευδοκώδικα στυλ C) η οποία επιστρέφει τον επόμενο χαρακτήρα από το πληκτρολόγιο αυτό. Όπως λέει και το όνομά της, η διαδικασία αυτή θα κάνει "busy wait", δηλαδή θα περιμένει να έλθει ο επόμενος χαρακτήρας απασχολώντας εν τω μεταξύ τον επεξεργαστή με το να ελέγχει συνεχώς, ξανά και ξανά, εάν ήλθε χαρακτήρας (ανάγνωση του καταχωρητή κατάστασης) --φυσικά, πρόκειται για πολύ κακό στυλ προγραμματισμού, αλλά από κάπου πρέπει να ξεκινήσουμε... Η διαδικασία θα επιστρέφει τον χαρακτήρα (char) που ήλθε. Θεωρήστε ότι η διαδικασία θα τρέχει σε kernel mode (θα είναι μέρος του λειτουργικού συστήματος), και ότι, όταν θα τρέχει, η μετάφραση εικονικών διευθύνσεων σε φυσικές θα είναι η συνάρτηση ταυτότητας, δηλαδή η φυσική διεύθυνση θα ισούται με την εικονική που την γέννησε. Χρησιμοποιήστε type casting, από τις σταθερές ακέραιες ποσότητες των διευθύνσεων που ξέρετε, για να αρχικοποιήσετε τους pointers (κατάλληλου είδους) που θα χρειαστείτε για προσπέλαση στους καταχωρητές της συσκευής.

(β) Η παραπάνω διαδικασία (α) είναι πολύ κακιά, διότι δεν αφήνει τον επεξεργαστή να κάνει τίποτα άλλο όσην ώρα αυτός περιμένει να πληκτρολογηθεί ο επόμενος χαρακτήρας. Όπως είπαμε και στο μάθημα, ένας καλύτερος τρόπος είναι να εκτελεί ο επεξεργαστής διάφορα προγράμματα, και, περιοδικά, όποτε έρχεται διακοπή από το ρολοί πραγματικού χρόνου (συνήθως 50 με 120 Hz --άλλο από το ρολοί του επεξεργαστή, των GHz ή εκατοντάδων MHz), μεταξύ άλλων περιοδικών εργασιών, να ελέγχει και εάν ήλθε κάποιος νέος χαρακτήρας από το πληκτρολόγιο (αρκεί οι χαρακτήρες να μην έρχονται πιά γρήγορα από τις διακοπές, πράγμα που ισχύει για

πληκτρολόγιο και 50-120 Hz ρυθμό διακοπών). Ο τρόπος αυτός λέγεται **δειγματοληψία (polling)** (ή περιόδευση), διότι ο επεξεργαστής παίρνει ένα "δείγμα" από την κατάσταση του πληκτρολογίου κάθε 10 με 20 ms (50-100 Hz). Γράψτε μία νέα διαδικασία `read_kbd_polling_char()`, ανάλογη με την προηγούμενη, αλλά αυτή τη φορά χωρίς αναμονή. Εάν έχει έλθει νέος χαρακτήρας από το προηγούμενο κάλεσμα στην `read_kbd_polling_char()`, τότε θα επιστρέφει αυτόν τον χαρακτήρα, αλλιώς (αν δεν έχει έλθει νέος χαρακτήρας) θα επιστρέφει (αμέσως) `'\0'`.

Άσκηση 12.3: Κόστος E/E βάσει Δειγματοληψίας και βάσει Διακοπών

Η περιοδική **δειγματοληψία (polling)** που είδαμε στην παραπάνω άσκηση 12.2 είναι ένας ρεαλιστικός τρόπος εισόδου/εξόδου (E/E - I/O), αρκεί η συχνότητα δειγματοληψίας να είναι αρκούντως ψηλή ώστε να μην χάνονται εισοδοί ή να μην καθυστερεί η έξοδος. Το μειονέκτημα της δειγματοληψίας είναι η σπατάλη χρόνου για την ανάγνωση του καταχωρητή κατάστασης όταν δεν έχει έλθει ακόμα νέα εισοδος ή δεν έχει τελειώσει ακόμα η προηγούμενη πράξη εξόδου.

Ένας εναλλακτικός τρόπος εισόδου/εξόδου είναι **E/E βάσει διακοπών (interrupt-driven I/O)**: η περιφερειακή συσκευή διακόπτει (interrupt) τον επεξεργαστή όταν υπάρχουν νέα δεδομένα εισόδου γι' αυτόν, ή όταν είναι έτοιμη να δεχτεί νέα δεδομένα εξόδου από αυτόν. Έτσι, δεν σπαταλιέται χρόνος για δειγματοληψία χωρίς λόγο της συσκευής, όσο αυτή δεν είναι ακόμα έτοιμη. Το κόστος, πάντως, της E/E βάσει διακοπών είναι η ειδική φροντίδα (overhead) που απαιτεί η κάθε διακοπή, δεδομένου ότι αυτή αλλάζει τη διεργασία που τρέχει, τα περιεχόμενα της κρυφής μνήμης και του TLB, και απαιτεί δαπανηρή καταγραφή στοιχείων (book-keeping) για να λειτουργήσει σωστά. Αντ' αυτού, η δειγματοληψία μπορεί να έχει το πλεονέκτημα, ανάλογα με την περίπτωση, ότι δειγματοληπτει "μία και καλή" πολλές συσκευές E/E για κάθε μία διακοπή από το ρολοί (batch processing), αντί να υφίσταται "κάθε τρεις και λίγο" το κόστος μίας επιπλέον διακοπής από μίαν άλλη συσκευή. Για να αποφασίσουμε τι μας συμφέρει μας ενδιαφέρουν τρεις παράμετροι:

- Πόσο κοντά χρονικά μπορεί να συμβούν δύο γεγονότα εισόδου; Η περίοδος δειγματοληψίας πρέπει να είναι βραχύτερη από αυτό, προκειμένου να μην χάσουμε το δεύτερο γεγονός. Προκειμένου περί εξόδου, πόσο δεχόμαστε να καθυστερήσουμε από την ολοκλήρωση μίας πράξης εξόδου μέχρι να το αντιληφθεί ο επεξεργαστής και να προχωρήσει στην επόμενη; Την παράμετρο αυτή μπορούμε να την μεγαλώσουμε (άρα σπανιότερη δειγματοληψία) αν η συσκευή E/E έχει έναν μεγαλύτερο ενταμειυτή (buffer) που να μπορεί να κρατά μέσα του περισσότερα δεδομένα εισόδου ή εξόδου (περισσότερη δουλειά κάθε φορά).
- Πόσος είναι ο μέσος ρυθμός των γεγονότων εισόδου; Δηλαδή, ανεξάρτητα αν δύο γεγονότα εισόδου ενδέχεται να συμβούν πολύ κοντά μεταξύ τους, *κατά μέσον όρο* πόσο κοντά χρονικά θα συμβαίνουν; Όσο σπανιότερη είναι η κατά μέσον όρο εμφάνισή τους, τόσο μεγαλύτερη είναι η σπατάλη της άσκοπης δειγματοληψίας.
- Πόσο πολλές συσκευές E/E δειγματοληπούμε μαζί σε κάθε διακοπή του ρολογιού (batching factor); Όσο περισσότερες είναι αυτές, τόσο περισσότερο αποσβένεται μεταξύ τους το κόστος της διακοπής του ρολογιού.

Θεωρήστε, σε αυτήν την άσκηση, ότι το ρολοί του επεξεργαστή είναι 1 GHz (άσχετο με το ρολοί πραγματικού χρόνου που μας δίνει περιοδικές διακοπές), ότι η ειδική φροντίδα (overhead) για κάθε διακοπή είναι δύο χιλιάδες (2000) κύκλοι του ρολογιού του επεξεργαστή, και ότι το κόστος δειγματοληψίας μίας συσκευής E/E είναι διακόσιοι (200) κύκλοι του ρολογιού του επεξεργαστή (η κύρια αιτία αυτής της καθυστέρησης είναι το ότι οι αρτηρίες E/E (I/O buses) είναι πολύ πιο αργές από τους (γρήγορους) σημερινούς επεξεργαστές). Θέλουμε να υπολογίσουμε τι ποσοστό του συνολικού χρόνου του επεξεργαστή θα απορροφά η E/E στις παρακάτω περιπτώσεις, όταν αυτή γίνεται βάσει δειγματοληψίας ή βάσει διακοπών.

(α) Έστω ένας υπολογιστής ο οποίος λαμβάνει και καταγράφει σήματα από 40 απομακρυσμένα σημεία. Κάθε μία από τις 40 γραμμές εισόδους ενδέχεται να φέρνει νέες εισόδους κάθε 1 ms, δηλαδή με μέγιστο ρυθμό 1 kHz. Εάν χρησιμοποιήσουμε δειγματοληψία, επομένως, το ρολόι πρέπει να μας δίνει 1 διακοπή ανά 1 ms. Σε κάθε διακοπή, δειγματοληπούμε 40 συσκευές. Πόσους κύκλους ρολογιού (του επεξεργαστή) ξοδεύουμε σε κάθε διακοπή, (i) για την ίδια τη διακοπή, και (ii) για τις 40 δειγματοληψίες; Δεδομένου ότι αυτό επαναλαμβάνεται 1000 φορές το δευτερόλεπτο, πόσους κύκλους ρολογιού ανά s ξοδεύουμε για E/E; Τι ποσοστό της συνολικής υπολογιστικής δυναμικότητας του επεξεργαστή αντιπροσωπεύουν αυτοί οι κύκλοι;

(β) Έστω ότι στο σύστημα (α), παρ' ότι νέες εισοδοί μπορεί να έρχονται σχετικά κοντά η μία με την άλλη (κάθε 1 ms), ο μέσος ρυθμός άφιξης τους είναι σημαντικά αραιότερος: κατά μέσον όρο έρχονται 20 νέες εισοδοί ανά δευτερόλεπτο ανά γραμμή εισόδου. Συνολικά, για όλες τις γραμμές, πόσες είναι οι νέες εισοδοί ανά s; Έστω ότι κάνουμε E/E βάσει διακοπών, και ότι κάθε νέα εισοδος (από οιαδήποτε γραμμή) προκαλεί μία διακοπή. Πόσες διακοπές ανά δευτερόλεπτο θα έχουμε, κατά μέσον όρο; Πόσους κύκλους ρολογιού θα ξοδεύει ο επεξεργαστής για να τις εξυπηρετήσει; Τι ποσοστό της συνολικής υπολογιστικής του δυναμικότητας αντιπροσωπεύουν αυτοί; Συμφέρει η δειγματοληψία (α) ή οι διακοπές (β);

(γ) Έστω τώρα ότι στο σύστημα (α) αυξάνεται ο μέσος ρυθμός άφιξης νέων εισόδων, από 20 ανά γραμμή ανά δευτερόλεπτο που ήταν στο (β) σε 400 ανά γραμμή ανά δευτερόλεπτο (δηλαδή πλησιάζει περισσότερο στο μέγιστο ρυθμό, που είναι 1 kHz). Το κόστος της δειγματοληψίας δεν αλλάζει, αφού αυτή ούτως ή άλλως επισκέπτεται την κάθε γραμμή 1000 φορές το δευτερόλεπτο. Όμως, στη μέθοδο βάσει διακοπών, αυξάνει το μέσο πλήθος διακοπών ανά δευτερόλεπτο. Πώς αλλάζουν οι απαντήσεις σας της ερώτησης (β) εδώ; Συμφέρει η δειγματοληψία ή οι διακοπές, τώρα;

(δ) Στις περιπτώσεις (β) και (γ), κινδυνεύουμε να χάσουμε κάποια νέα εισοδος αν "πέσουν μαζεμένες" νέες εισοδοί από όλες τις γραμμές; Έστω ότι αμέσως μετά την έλευση μίας νέας εισόδου από τη γραμμή A, μας έρχονται νέες εισοδοί και από τις 39 άλλες γραμμές. Για να εξυπηρετήσει ο επεξεργαστής τις 40 αυτές διακοπές (τη μία μετά την άλλη), πόσους κύκλους επεξεργαστή χρειάζεται; Πόσος χρόνος είναι αυτός; Το νωρίτερο που μπορεί να έλθει η επόμενη εισοδος από τη γραμμή A είναι 1 ms μετά την προηγούμενη, όπως είπαμε στο (α). Θα έχει προλάβει ο επεξεργαστής να εξυπηρετήσει τις παραπάνω 40 διακοπές πριν έλθει η επόμενη αυτή εισοδος από τη γραμμή A, ή θα την χάσει αυτή την επόμενη εισοδος;

(ε) Έστω τώρα ότι αντί των 40 εισόδων του (α) ο υπολογιστής μας έχει μία μόνο --αλλά πολύ γρηγορότερη-- εισοδος: μία γραμμή δικτύου των 32 Mbit/s, δηλαδή 4 MBytes/s. Έστω ότι η συσκευή εισόδου μπορεί να κρατήσει (έχει buffer για να κρατήσει) 1 πακέτο, αλλά όχι παραπάνω. Η συσκευή μας δίνει 1 διακοπή για κάθε 1 αφικνούμενο πακέτο. Έστω ότι τα μικρότερα δυνατά πακέτα είναι μεγέθους 40 Bytes καθένα (όπως στο πρωτόκολλο του Διαδικτύου, το IP). Άρα, ο μέγιστος δυνατός ρυθμός άφιξης πακέτων είναι 4000 KBytes/s διά 40 Bytes ανά πακέτο = 100 K πακέτα/s. Εάν μας έρχονται πακέτα ελαχίστου μεγέθους και με τον μέγιστο δυνατό ρυθμό ("κολλητά" το ένα με το άλλο), ξανα-απαντήστε την ερώτηση (β).

(στ) Σήμερα υπάρχουν γραμμές δικτύου π.χ. του 1 Gbit/s, δηλαδή περίπου 30 φορές γρηγορότερες από αυτήν του (ε). Μπορεί ο υπολογιστής μας να τις αντέξει αν η συσκευή εισόδου συνεχίσει να έχει ενταμειυτή μόνο για ένα πακέτο και συνεχίσει να μας δίνει μία διακοπή για κάθε αφικνούμενο πακέτο;;

Άσκηση 12.4: Απευθείας Πρόσβαση Μνήμης (DMA) από Συσκευές E/E

Από τα παραπάνω νούμερα φάνηκε ότι οι γρήγορες συσκευές E/E πρέπει να έχουν μεγάλους ενταμειυτές (buffers), ούτως ώστε οι διακοπές --είτε του ρολογιού είτε των συσκευών-- να μην είναι πολύ συχνές, και να μπορεί μεγάλη "ποσότητα εργασίας" να συσσωρεύεται στον ενταμειυτή μεταξύ διαδοχικών "επισκέψεων" στη συσκευή από τον επεξεργαστή. Ακόμα και με αυτή τη λύση, όμως, για να μην είναι οι διακοπές πολύ

συχνές, υπάρχει και ένα άλλο πρόβλημα επιδόσεων για τις γρήγορες συσκευές E/E:

Γιά να αντιγράψει ο επεξεργαστής ένα μεγάλο όγκο δεδομένων ανάμεσα στον ενταμειυτή της περιφερειακής συσκευής και την κυρίως μνήμη του υπολογιστή, απαιτούνται πολλοί κύκλοι ρολογιού, επειδή οι αρτηρίες E/E (λεωφόροι E/E - I/O buses) είναι πολύ πιο αργές από τους σημερινούς (γρήγορους) επεξεργαστές. Δεδομένου ότι η αντιγραφή αυτή είναι μία πολύ απλή εργασία, θα αποτελούσε σπατάλη δαπανηρών υπολογιστικών πόρων (του επεξεργαστή) το να βάζουμε τον επεξεργαστή να την κάνει: ο επεξεργαστής, σε αυτή τη δουλειά, θα σπαταλά την περισσότερη ώρα του περιμένοντας να απαντήσει η αρτηρία E/E. Η ενδεδειγμένη λύση είναι να αποκτήσει η περιφερειακή συσκευή τη δυνατότητα να κάνει μόνη της την αντιγραφή ανάμεσα στο ενταμειυτή της και στην κύρια μνήμη: Η "**Απευθείας Πρόσβαση Μνήμης (Direct Memory Access - DMA)**" από τις συσκευές E/E λειτουργεί ως εξής. Η συσκευή E/E έχει δύο ή τρεις καταχωρητές ελέγχου για τη λειτουργία DMA:

- Διεύθυνση έναρξης --είναι η (φυσική) διεύθυνση μνήμης προς την οποία ή από την οποία θα αρχίσει η αντιγραφή δεδομένων.
- Μέγεθος μεταφοράς --είναι το πλήθος των Bytes που θα αντιγραφούν.
- Καταχωρητής ενεργοποίησης --είναι ο καταχωρητής εκείνος όπου μόλις ο επεξεργαστής γράψει έναν ειδικό κώδικα θα αρχίσει η αντιγραφή (αυτός μπορεί και να μην υπάρχει, π.χ. εάν η αντιγραφή αρχίζει με το που γράφουμε το μέγεθος μεταφοράς στον δεύτερο καταχωρητή).

Η συσκευή E/E έχει επίσης μία μικρή μηχανή πεπερασμένων καταστάσεων (FSM), η οποία, μόλις δοθεί το σήμα εκκίνησης, κάνει την εξής δουλειά κατ' επανάληψη:

- Ζητά να της δοθεί η χρήση της αρτηρίας μνήμης (ή των αρτηριών E/E και μνήμης, ή του/των σχετικών συνδέσμων).
- Μόλις της δοθεί η χρήση, αντιγράφει την επόμενη "λέξη" του ενταμειυτή της συσκευής στην κυρίως μνήμη, εκεί που δείχνει ο καταχωρητής διεύθυνσης, ή την αντιγράφει από την κυρίως μνήμη στον ενταμειυτή της συσκευής. [Στο βήμα αυτό, η "λέξη" που αντιγράφεται συχνά δεν είναι μία μόνο λέξη του επεξεργαστή ή της αρτηρίας, αλλά μία μικρή ομάδα (burst - ριπή) λέξεων, προκειμένου να εκμεταλλευτούμε τη δυνατότητα των DRAM για οικονομικότερη προσπέλαση συνεχόμενων λέξεων (§ 12.0), καθώς και να αποσβέσουμε καλύτερα το overhead απόκτησης χρήσης της αρτηρίας μέσω της μεταφοράς περισσότερων Bytes κάθε φορά που την αποκτούμε].
- Αυξάνει τον καταχωρητή διεύθυνσης κατά το πλήθος των Bytes που μόλις μετέφερε.
- Μειώνει τον καταχωρητή μεγέθους μεταφοράς κατά το πλήθος των Bytes που μόλις μετέφερε.
- Αν ο καταχωρητής μεγέθους είναι ακόμα μεγαλύτερος του μηδενός, επαναλαμβάνει από το (i).

Ας θεωρήσουμε σε αυτή την άσκηση τον ίδιο επεξεργαστή με ρολοί 1 GHz που είχαμε και παραπάνω, και έναν σύνδεσμο επικοινωνίας περιφερειακών - μνήμης - επεξεργαστή τύπου PCI-Express των 2.5 Gbits/s (κάπως παλαιότερης γενιάς), και "στενό", δηλ. της μίας μόνο "λωρίδας" (x1). Ο ρυθμός μετάδοσης αυτού του συνδέσμου είναι $2.5 \text{ Gb/s} = 2500 \text{ Mb/s} = \text{περίπου } 312 \text{ Mbytes/s}$, δηλαδή περίπου 1 Byte κάθε 3.2 ns ή αλλιώς 1 Byte κάθε 3.2 κύκλοι ρολογιού του επεξεργαστή μας. Πάνω στο σύνδεσμο αυτό οι πληροφορίες μεταφέρονται σε "πακέτα" (όπως στα δίκτυα), και κάθε πακέτο έχει μίαν επικεφαλίδα (header) ως πούμε των 12 Bytes (π.χ. διευθύνσεις πηγής, προορισμού, κωδικό πράξης, έλεγχος σφαλμάτων, κλπ) ακολουθούμενη από τα μεταφερόμενα Bytes δεδομένων.

(α) Έστω ότι δεν υπάρχει DMA, και ο επεξεργαστής κάνει την αντιγραφή μεταξύ μνήμης και ενταμειυτή περιφερειακής συσκευής, ας πούμε από τη μνήμη προς την περιφερειακή συσκευή. Η αντιγραφή γίνεται με ένα μικρό βρόχο που περιλαμβάνει μία εντολή load από τη μνήμη και μία εντολή store στον ενταμειυτή της συσκευής. Η εντολές load και store αναφέρονται σε μία μόνο λέξη (4 Bytes) --αφού δεν υπάρχουν εντολές load/store πολλαπλών λέξεων. Όταν ταξιδεύει η λέξη (μέσω της εντολής store) προς τη

συσκευή μέσω του σύνδεσμου PCI-Express, η λέξη αυτή ταξιδεύει μέσα σε ένα πακέτο μεγέθους 12 Bytes (header) + 4 Bytes (data) = 16 Bytes ανά πακέτο. Με το ρυθμό 1 Byte κάθε 3.2 κύκλοι ρολογιού του επεξεργαστή που βρήκαμε παραπάνω, κάθε τέτοιο πακέτο των 16 Bytes (μεταφορά μίας λέξης των 4 Bytes) κοστίζει περίπου 50 κύκλους ρολογιού του επεξεργαστή. Ας υποθέσουμε ότι οι υπόλοιπες εντολές του βρόχου κοστίζουν 30 κύκλους του επεξεργαστή, κυρίως λόγω των αναπόφευκτων αστοχιών κρυφής μνήμης που θα προκαλέσουν οι επανειλημμένες εντολές load από διευθύνσεις μη πρόσφατα χρησιμοποιημένες. Συνολικά, επομένως, ο ρυθμός αντιγραφής είναι 4 Bytes ανά 80 κύκλους επεξεργαστή (80 ns). Πόσος είναι αυτός ο ρυθμός σε MBytes/s και σε Mbits/s; Εάν ο επεξεργαστής αυτός έχει να τροφοδοτεί ταυτόχρονα 2 δίσκους με παροχή 10 MBytes/s καθένας και 1 δίκτυο fast ethernet με παροχή 100 Mbits/s, τι ποσοστό του χρόνου του θα υποχρεωθεί να αφιερώνει για αντιγραφές δεδομένων από τη μνήμη του προς τους ενταμειντές των συσκευών αυτών;

(β) Έστω τώρα ότι υπάρχει DMA. Ας υποθέσουμε, προς στιγμήν, ότι ο επεξεργαστής δεν απασχολεί καθόλου το σύνδεσμο PCI-Express μνήμης-E/E, π.χ. επειδή ευστοχεί συνεχώς στην κρυφή του μνήμη, και επομένως ο σύνδεσμος αυτός είναι συνεχώς διαθέσιμος για τη μηχανή DMA. (i) Εάν η μηχανή DMA τροφοδοτούσε τις περιφερειακές συσκευές στέλλοντας μία λέξη κάθε φορά (4 data Bytes μέσα σε κάθε "πακέτο"), και εάν απασχολεί έτσι το σύνδεσμο πλήρως (100%), πόση θα ήταν η συνολική παροχή της σε MBytes/s και σε Mbits/s; (ii) Όμως, οι μηχανές DMA κάνουν (σχεδόν) πάντα τις μεταφορές τους σε μεγαλύτερα blocks δεδομένων, προκειμένου να αποσβέσουν σε περισσότερα δεδομένα το κόστος (overhead) της επικεφαλίδας του πακέτου, ή της απόκτησης και χρήσης της αρτηρίας, ή της έναρξης προσπέλασης σε νέα γραμμή (row) της μνήμης DRAM. Ας πούμε ότι εδώ η μηχανή DMA μας μεταφέρει τα δεδομένα σε ομάδες (blocks) των 16 λέξεων (64 Bytes) καθεμία (δηλ. ανά πακέτο): άρα τα πακέτα θα έχουν μέγεθος 12 header Bytes + 64 data Bytes = 76 Bytes κάθε πακέτο. Με το ρυθμό 1 Byte κάθε 3.2 ns που βρήκαμε παραπάνω, κάθε τέτοιο πακέτο παίρνει $76 \text{ B} \times 3.2 \text{ ns/B} =$ περίπου 250 ns ανά πακέτο, ή 4 M πακέτα (των 64 Bytes δεδομένων) ανά δευτερόλεπτο. Εάν η αρτηρία απασχολείται πάλι 100% για τέτοιες μεταφορές DMA, πόση θα είναι η συνολική παροχή της σε MBytes/s και σε Mbits/s;

(γ) Εάν οι παραπάνω μεταφορές DMA σε blocks των 64 Bytes εξυπηρετούν πάλι τους 2 δίσκους με παροχή 10 MBytes/s καθένας και το 1 δίκτυο fast ethernet με παροχή 100 Mbits/s της ερώτησης (α), τότε τι ποσοστό του χρόνου του συνδέσμου PCI-Express μνήμης-E/E απασχολούν αυτές οι μεταφορές DMA αυτών των περιφερειακών συσκευών; Συγκρίνετε αυτό το ποσοστό με το ποσοστό της απάντησης (α). Παρ'ότι πρόκειται για ανόμοια μεγέθη (το (α) ήταν ποσοστό του χρόνου του επεξεργαστή, ενώ το (γ) είναι ποσοστό του χρόνου της αρτηρίας), εξηγήστε σε ποιούς δύο παράγοντες οφείλεται η μείωση του ποσοστού απασχόλησης από το (α) στο (γ).

Επιπλέον της μείωσης αυτής, που είναι από μόνη της ένα κέρδος, παρατηρήστε ότι στο μεν (α), δηλαδή χωρίς DMA, ο επεξεργαστής αφιέρωνε ένα μη ευκαταφρόνητο μέρος του χρόνου του για να εξυπηρετεί τις μεταφορές δεδομένων αυτών των συσκευών E/E, ενώ στο (γ), δηλαδή με DMA, ο επεξεργαστής δεν αφιερώνει καθόλου χρόνο σε αυτές τις μεταφορές (φροντίζει μόνο να τις ξεκινάει, και μετά τις αφήνει να τρέχουν μόνες τους για πολλά KBytes συνήθως), και οι μεταφορές γίνονται "μόνες τους" (δηλαδή από την ή τις μηχανές DMA, που δουλεύουν παράλληλα με τον επεξεργαστή), απασχολώντας (οι μεταφορές DMA) ένα μέρος μόνο της διαθέσιμης παροχής (throughput) της μνήμης, και αφήνοντας το υπόλοιπο μέρος αυτής της παροχής διαθέσιμο για να εξυπηρετούνται οι αστοχίες της κρυφής μνήμης του επεξεργαστή.