Σειρά Ασκήσεων 17: Μονάδες Εισόδου/Εξόδου (I/O), Αρτηρίες (Buses), DMA

Προθεσμία έως Κυριακή 23 Μαΐου 2010 (εβδομάδα 13)

[**Up** - Table of Contents] [printer version - **PDF**]

Άσκηση 17.1: Απεικόνιση Μνήμης των Μονάδων Ε/Ε (Memory Mapped I/O)

Οπως είπαμε στο μάθημα, ένας συνηθισμένος τρόπος επικοινωνίας επεξεργαστή-μονάδων εισόδου/εξόδου (Ε/Ε - περιφερειακές συσκευές) είναι η "απεικόνιση μνήμης" των μονάδων Ε/Ε (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 εξσφαλίζεται 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 Ε/Ε που βλέπει φυσικές διευθύνσεις λέξεων (όχι bytes, δήλαδή έχουν ήδη αφαιρεθεί τα 2-3 LS bits της διεύθυνσης του επεξεργαστή) μεγέθους (οι φυσικές διευθύνσεις λέξεων) 16 bits. Τον αντίστοιχο χώρο φυσικών διευθύνσεων, μεγέθους 65536 λέξεων, αποφασίζουμε να μοιράσουμε ως εξής:

- **`0xxxxxxxxxxxxxxx:`** 32768 λέξεις κύριας μνήμης (main memory).
- **`10xxxxxxxxxxxxxx:`** 16384 λέξεις για μία "μεγάλη" περιφερειακή συσκευή (δηλαδή μία συσκευή που περιέχει έναν μεγάλο buffer δεδομένων Ε/Ε που θέλουμε να μπορεί να βλέπει ο επεξεργαστής).
- **`110xxxxxxxxxxxxx, 1110xxxxxxxxxxxx:`** 12288 λέξεις κατανεμημένες σε 3 "μεσαίες" συσκευές Ε/Ε, όπου η κάθε μία από αυτές τις συσκευές μπορεί να χρησιμοποιεί έως 4096 διαφορετικές διευθύνσεις για εσωτερικούς της σκοπούς.
- **`1111xxxxxxxxxxxx:`** 4096 λέξεις κατανεμημένες σε έως 32 "μικρές" συσκευές Ε/Ε, όπου η κάθε μία από αυτές τις συσκευές μπορεί να χρησιμοποιεί έως 128 διαφορετικές διευθύνσεις για να επιλέγει εσωτερικούς της καταχωρητές ή ειδικές πράξεις/λειτουργίες.

(α) Σχεδιάστε, χρησιμοποιώντας πύλες AND και NOT, τον αποκωδικοποιητή διευθύνσεων που επιλέγει τη συσκευή που πρέπει να ενεργοποιηθεί κάθε φορά. Είσοδος του αποκωδικοποιητή είναι τα 16 σύρματα φυσικής διεύθυνσης λέξεων από τον επεξεργαστή (μετά τη μετάφραση της εικονικής διεύθυνσης από το TLB) –ή όσα από αυτά χρειάζεστε. Ο αποκωδικοποιητής έχει 37 σύρματα εξόδου: 1 για την κύρια μνήμη, 1 για την μεγάλη συσκευή Ε/Ε, 3 για τις μεσαίες συσκευές Ε/Ε, και 32 για τις μικρές συσκευές Ε/Ε. Εσείς σχεδιάστε το κύκλωμα που γεννά τις πρώτες 12 από αυτές τις 37 εξόδους, δείχνοντας προσεκτικά ποια σύρματα εισόδου και ποιας πολικότητας χρησιμοποιείτε σε κάθε πύλη AND.

(β) Σε ποια λέξη μνήμης (στο δεκαδικό σύστημα, π.χ. #135 αρχίζοντας από την "#0") ή σε ποιον καταχωρητή (στο δεκαδικό σύστημα π.χ. "#5", αρχίζοντας από τον "#0", σημειώνοντας ότι οι καταχωρητές κάθε συσκευής απεικονίζονται στις διευθύνσεις μνήμης που αντιστοιχούν στη συσκευή ένας προς μία) ποιας συσκευής (στο δεκαδικό σύστημα π.χ. "μικρής #3", αρχίζοντας από την "μικρή #0") αναφέρεται κάθε μία από τις εξής φυσικές διευθύνσεις λέξεων που δίδονται στο δεκαεξαδικό σύστημα: A000, 0008, 0FFA, 3FF8, 6000, 7FFC, 8000, 8FF2, A000, B5FF, C600, C640, C680, C6C0, E100, E240, E360, E380, E4A0, E5C0, E6F0, E6F4, F7F8, F8FC, F9FF.

Άσκηση 17.2: Καταχωρητές Κατάστασης, Busy Wait, Polling

Φυσικά, οι μονάδες Ε/Ε 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 Ε/Ε, 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 Ε/Ε δεν συμπεριφέρονται σαν πραγματική μνήμη, οι τιμές που διαβάζουμε ή γράφουμε στις διευθύνσεις τους πρέπει να *μην* κρατιόνται στην *κρυφή* μνήμη, ειδάλως θα διαβάζουμε παλιές τιμές ή αυτά που γράφουμε δεν θα φτάνουν όλα ή αμέσως στις συσκευές Ε/Ε. Αυτό, το να παρακάμπτουν δηλαδή οι προσπελάσεις αυτές την κρυφή μνήμη, επιτυγχάνεται συνήθως με το να αναγνωρίζει η κρυφή μνήμη την ειδική μορφή των φυσικών διευθύνσεων των συσκευών Ε/Ε.

Ένα άλλο σύστημα επικοινωνίας επεξεργαστή-συσκευών Ε/Ε, διαφορετικό από την απεικόνιση μνήμης των μονάδων Ε/Ε, 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 Ε/Ε είναι "προνομιούχες" (priviledged), δηλαδή επιτρέπεται να εκτελούνται μόνο σε "kernel mode", και οι εντολές Ε/Ε ειδοποιούν την κρυφή μνήμη να μην παρέμβει. Κατά τα άλλα, στις αρτηρίες Ε/Ε, οι εντολές Ε/Ε μάλλον καταλήγει να δίνουν διευθύνσεις εντελώς ανάλογες προς αυτές που δίνουν οι εντολές load/store στα συστήματα με απεικόνιση μνήμης των μονάδων Ε/Ε.

Σε αυτή την άσκηση, θεωρήστε ότι η "μικρή" συσκευή Ε/Ε #16 της άσκησης 17.1 είναι μία συσκευή εισόδου από πληκτρολόγιο, ότι ο καταχωρητής #0 αυτής της συσκευής είναι ο "καταχωρητής κατάστασης", και ότι ο καταχωρητής της #1 είναι ο "καταχωρητής δεδομένων". 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_s7_busywait_char()`**" σε 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) δίνει μία υλοποίηση κάκιστης επίδοσης και κάκιστης κατανάλωσης ενέργειας και πόρων του επεξεργαστή, 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 ρολόϊ πραγματικού χρόνου (συνήθως 50 με 100 Hz –άλλο από το ρολόϊ του επεξεργαστή, των πολλών εκατοντάδων MHz), μεταξύ άλλων περιοδικών εργασιών, να ελέγχει και εάν ήλθε κάποιος νέος χαρακτήρας από το πληκτρολόγιο (αρκεί οι χαρακτήρες να μην έρχονται πιό γρήγορα από τις διακοπές, πράγμα που ισχύει για πληκτρολόγιο και 50–100 Hz ρυθμό διακοπών). Ο τρόπος αυτός λέγεται **δειγματοληψία (polling)**, διότι ο επεξεργαστής παίρνει ένα "δείγμα" από την κατάσταση του πληκτρολογίου κάθε 10 με 20 ms (50–100 Hz). Γράψτε μία νέα διαδικασία "**`read_s7_polling_char()`**", ανάλογη με την προηγούμενη, αλλά αυτή τη φορά χωρίς αναμονή. Εάν έχει έλθει νέος χαρακτήρας από το προηγούμενο κάλεσμα στην read_s7_polling_char(), τότε θα επιστρέφει αυτόν τον χαρακτήρα, αλλοιώς (αν δεν έχει έλθει νέος χαρακτήρας) θα επιστρέφει (αμέσως) **`'\0'`**.

Άσκηση 17.3: Κόστος Ε/Ε βάσει Δειγματοληψίας και βάσει Διακοπών

Η περιοδική **δειγματοληψία (polling)** που είδαμε στην παραπάνω άσκηση 17.2 είναι ένας ρεαλιστικός τρόπος εισόδου/εξόδου (Ε/Ε - I/O), αρκεί η συχνότητα δειγματοληψίας να είναι αρκούντως ψηλή ώστε να μην χάνονται είσοδοι 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 είσοδος ή δεν έχει τελειώσει ακόμα η προηγούμενη πράξη εξόδου.

Ένας εναλλακτικός τρόπος εισόδου/εξόδου είναι **Ε/Ε 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 Ε/Ε 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 δειγματοληπτεί "μία και καλή" πολλές συσκευές Ε/Ε για κάθε μία διακοπή από το ρολόϊ (batch processing), αντί να υφίσταται "κάθε τρείς και λίγο" το κόστος μίας επιπλέον διακοπής από μίαν άλλη συσκευή. Γιά 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 Ε/Ε έχει έναν μεγαλύτερο ενταμιευτή (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; Οσο σπανιότερη είναι η κατά μέσον όρο εμφάνισή τους, τόσο μεγαλύτερη είναι η σπατάλη της άσκοπης δειγματοληψίας. Προκειμένου περί εξόδου, πόσο περισσότερο διαρκεί κατά μέσον όρο η κάθε πράξη από την περίοδο δειγματοληψίας (δηλ. την μέγιστη αποδεκτή καθυστέρηση αντίδρασης);
- Πόσο πολλές συσκευές Ε/Ε δειγματοληπτούμε μαζί σε κάθε διακοπή του ρολογιού (batching factor); Οσο περισσότερες είναι αυτές, τόσο περισσότερο αποσβένεται μεταξύ τους το κόστος της διακοπής του ρολογιού.

Θεωρήστε, σε αυτήν την άσκηση, ότι το ρολόϊ του επεξεργαστή είναι 1 GHz (άσχετο με το ρολόϊ πραγματικού χρόνου που μας δίνει περιοδικές διακοπές), ότι η ειδική φροντίδα (overhead) για κάθε διακοπή είναι πέντε χιλιάδες (5000) κύκλοι του ρολογιού του επεξεργαστή, και ότι το κόστος δειγματοληψίας μίας συσκευής Ε/Ε είναι πεντακόσιοι (500) κύκλοι του ρολογιού του επεξεργαστή (η κύρια αιτία αυτής της καθυστέρησης είναι το ότι οι αρτηρίες Ε/Ε (I/O buses) είναι πολύ πιό αργές από τους (γρήγορους) σημερινούς επεξεργαστές). Θέλουμε να υπολογίσουμε τι ποσοστό του συνολικού χρόνου του επεξεργαστή θα απορροφά η Ε/Ε στις παρακάτω περιπτώσεις, όταν αυτή γίνεται βάσει δειγματοληψίας ή βάσει διακοπών.

(α) Εστω ένας υπολογιστής ο οποίος λαμβάνει και καταγράφει σήματα από 100 απομακρυσμένα σημεία. Κάθε μία από τις 100 γραμμές εισόδους ενδέχεται να φέρνει νέες εισόδους κάθε 1 ms, δηλαδή με μέγιστο ρυθμό 1 KHz. Εαν χρησιμοποιήσουμε δειγματοληψία, επομένως, το ρολόϊ πρέπει να μας δίνει 1 διακοπή ανά 1 ms. Σε κάθε διακοπή, δειγματοληπτούμε 100 συσκευές. Πόσους κύκλους ρολογιού (του επεξεργαστή) ξοδεύουμε σε κάθε διακοπή, (i) για την ίδια τη διακοπή, και (ii) για τις 100 δειγματοληψίες; Δεδομένου ότι αυτό επαναλαμβάνεται 1000 φορές το δευτερόλεπτο, πόσους κύκλους ρολογιού ανά s ξοδεύουμε για Ε/Ε; Τι ποσοστό της συνολικής υπολογιστικής δυναμικότητας του επεξεργαστή αντιπροσωπεύουν αυτοί οι κύκλοι;

(β) Έστω ότι στο σύστημα (α), παρ' ότι νέες είσοδοι μπορεί να έρχονται σχετικά κοντά η μία με την άλλη (κάθε 1 ms), ο μέσος ρυθμός άφιξής τους είναι σημαντικά αραιότερος: κατά μέσον όρο έρχονται 50 νέες είσοδοι ανά δευτερόλεπτο ανά γραμμή εισόδου. Συνολικά, για όλες τις γραμμές, πόσες είναι οι νέες είσοδοι ανά s; Έστω ότι κάνουμε Ε/Ε βάσει διακοπών, και ότι κάθε νέα είσοδος (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 50 ανά γραμμή ανά δευτερόλεπτο που ήταν στο (β) σε 500 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 είσοδο αν "πέσουν μαζεμένες" νέες είσοδοι από όλες τις γραμμές; Έστω ότι αμέσως μετά την έλευση μίας νέας εισόδου από τη γραμμή Α, μας έρχονται νέες είσοδοι και από τις 99 άλλες γραμμές. Γιά να εξυπηρετήσει ο επεξεργαστής τις 100 αυτές διακοπές (τη μία μετά την άλλη), πόσους κύκλους επεξεργαστή χρειάζεται; Πόσος χρόνος είναι αυτός; Το νωρίτερο που μπορεί να έλθει η επόμενη είσοδος από τη γραμμή Α είναι 1 ms μετά την προηγούμενη, όπως είπαμε στο (α). Θα έχει προλάβει ο επεξεργαστής να εξυπηρετήσει τις παραπάνω 100 διακοπές πριν έλθει η επόμενη αυτή είσοδος από τη γραμμή Α, ή θα την χάσει αυτή την επόμενη είσοδο;

(ε) Έστω τώρα ότι αντί των 100 εισόδων του (α) ο υπολογιστής μας έχει δύο (2) εισόδους, αλλά αυτές είναι γρηγορότερες. Έστω ότι κάθε είσοδος είναι μία γραμμή δικτύου των 1 Gbit/s, δηλαδή περίπου 120 MBytes/s. Έστω ότι κάθε συσκευή εισόδου μπορεί να κρατήσει (έχει buffer για να κρατήσει) 1 πακέτο, αλλά όχι παραπάνω. Όταν κάνουμε Ε/Ε βάσει διακοπών, κάθε 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 120 MBytes/s διά 40 Bytes ανά πακέτο = 3 Μ πακέτα/s ανά γραμμή. Έστω, δε, ότι ο μέσος ρυθμός άφιξης πακέτων είναι 1 Μ πακέτα/s ανά γραμμή. Με αυτά τα νούμερα, ξανα-απαντήστε τις ερωτήσεις (α) και (β). Αποτελούν τώρα αυτές οι συσκευές Ε/Ε ελαφρύ φορτίο για τον υπολογιστή μας, όπως στις περιπτώσεις (α)-(γ), ή σημαντικό/βαρύ φορτίο; Μπορεί ο υπολογιστής μας να τις αντέξει αν η συσκευή εισόδου συνεχίσει να έχει ενταμιευτή μόνο για ένα πακέτο, ή συνεχίσει να μας δίνει μία διακοπή για κάθε αφικνούμενο πακέτο;;;

Άσκηση 17.4: Απευθείας Πρόσβαση Μνήμης (DMA) από Συσκευές Ε/Ε

Από τα παραπάνω νούμερα φάνηκε ότι οι γρήγορες συσκευές Ε/Ε πρέπει να έχουν μεγάλους ενταμιευτές (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 ενταμιευτή 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 Ε/Ε:

Γιά να αντιγράψει ο επεξεργαστής ένα μεγάλο όγκο δεδομένων ανάμεσα στον ενταμιευτή της περιφερειακής συσκευής και την κυρίως μνήμη του υπολογιστή, απαιτούνται πολλοί κύκλοι ρολογιού, επειδή οι αρτηρίες Ε/Ε (λεωφόροι Ε/Ε - I/O buses) είναι πολύ πιό 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 Ε/Ε. Η ενδεδειγμένη λύση είναι να αποκτήσει η περιφερειακή συσκευή τη δυνατότητα να κάνει μόνη της την αντιγραφή ανάμεσα στο ενταμιευτή της και στην κύρια μνήμη: Η "**Απευθείας Πρόσβαση Μνήμης (Direct Memory Access - DMA)**" από τις συσκευές Ε/Ε λειτουργεί ως εξής. Η συσκευή Ε/Ε έχει 3 καταχωρητές ελέγχου για τη λειτουργία DMA:

a. Διεύθυνση έναρξης –είναι η (φυσική) διεύθυνση μνήμης προς την οποία ή από την οποία θα αρχίσει η αντιγραφή δεδομένων.
b. Μέγεθος μεταφοράς –είναι το πλήθος των Bytes που θα αντιγραφούν.
c. Καταχωρητής ενεργοποίησης –είναι ο καταχωρητής εκείνος όπου μόλις ο επεξεργαστής γράψει έναν ειδικό κώδικα θα αρχίσει η αντιγραφή.

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

i. Ζητά να της δοθεί η χρήση της αρτηρίας μνήμης ή των αρτηριών Ε/Ε και μνήμης.
ii. Μόλις της δοθεί η χρήση, αντιγράφει την επόμενη "λέξη" του ενταμιευτή της συσκευής στην κυρίως μνήμη, εκεί που δείχνει ο καταχωρητής διεύθυνσης, ή την αντιγράφει από την κυρίως μνήμη στον ενταμιευτή 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) λέξεων, προκειμένου να εκμεταλλευτούμε τη δυνατότητα των DRAM για οικονομικότερη προσπέλαση συνεχόμενων λέξεων (§ **16.1**), καθώς και να αποσβέσουμε καλύτερα το overhead απόκτησης χρήσης της αρτηρίας μέσω της μεταφοράς περισσοτέρων Bytes κάθε φορά που την αποκτούμε].
iii. Αυξάνει τον καταχωρητή διεύθυνσης κατά το πλήθος των Bytes που μόλις μετέφερε.

iv. Μειώνει τον καταχωρητή μεγέθους μεταφοράς κατά το πλήθος των Bytes που μόλις μετέφερε.
v. Αν ο καταχωρητής μεγέθους είναι ακόμα μεγαλύτερος του μηδενός, επαναλαμβάνει από το (i).

Ας θεωρήσουμε σε αυτή την άσκηση τον ίδιο επεξεργαστή με ρολόϊ 1 GHz που είχαμε και παραπάνω, με μία ενιαία (για απλότητα) αρτηρία μνήμης-Ε/Ε, όπου η αρτηρία λειτουργεί με ρολόϊ 100 MHz (10 φορές πιό αργό). Η αρτηρία έχει πλάτος 64 bits = 8 Bytes. Κάθε χρήση της αρτηρίας κοστίζει: (i) 2 κύκλους της αρτηρίας (= 20 κύκλους επεξεργαστή) overhead για το ξεκίνημα (διαιτησία, έλεγχος, επιλογή συσκευής, μεταφορά διεύθυνσης), συν (ii) 1 επιπλέον κύκλο αρτηρίας (= 10 κύκλους επεξεργαστή) για κάθε 64 bits = 8 Bytes μεταφερομένων δεδομένων.

(α) Έστω ότι δεν υπάρχει DMA, και ο επεξεργαστής κάνει την αντιγραφή μεταξύ ενταμιευτή περιφερειακής συσκευής και μνήμης, ας πούμε από τη συσκευή προς τη μνήμη. Η αντιγραφή γίνεται με ένα μικρό βρόχο που περιλαμβάνει μία εντολή load από τη συσκευή και μία εντολή store στη μνήμη. Η εντολή load αναφέρεται σε μία μόνο λέξη (ας πούμε των 64 bits) –αφού δεν υπάρχουν εντολές load/store πολλαπλών λέξεων. Επειδή διαβάζει από την αρτηρία Ε/Ε και όχι από την (κρυφή) μνήμη, αυτή κοστίζει, κατά τα παραπάνω, 3 κύκλους της αρτηρίας (2 overhead εκκίνησης + 1 για τη μία λέξη δεδομένων) = 30 κύκλους του επεξεργαστή. Ας υποθέσουμε ότι οι υπόλοιπες εντολές του βρόχου κοστίζουν 10 κύκλους του επεξεργαστή, κυρίως λόγω των αναπόφευκτων αστοχιών κρυφής μνήμης που θα προκαλέσουν οι επανειλημμένες εντολές store σε διευθύνσεις μη πρόσφατα χρησιμοποιημένες. Συνολικά, επομένως, ο ρυθμός αντιγραφής είναι 64 bits = 8 Bytes ανά 40 κύκλους επεξεργαστή. Πόσος είναι αυτός ο ρυθμός σε MBytes/s και σε Mbits/s; Εάν ο επεξεργαστής αυτός έχει να εξυπηρετεί ταυτόχρονα 2 δίσκους με παροχή 40 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 τους ενταμιευτές των συσκευών αυτών προς τη μνήμη του;

(β) Έστω τώρα ότι υπάρχει DMA. Ας υποθέσουμε, προς στιγμήν, ότι ο επεξεργαστής δεν απασχολεί καθόλου την αρτηρία μνήμης-Ε/Ε, π.χ. επειδή ευστοχεί συνεχώς στην κρυφή του μνήμη, και επομένως η αρτηρία αυτή είναι συνεχώς διαθέσιμη στην (στις) συσκευή(ες) DMA. Έστω (i) ότι οι συσκευές DMA κάνουν τις αντιγραφές τους μέσω μεταφορών μίας (1) λέξης (των 64 bits = 8 Bytes) κάθε φορά, η οποία κοστίζει, κατά τα παραπάνω, 3 κύκλους της αρτηρίας η κάθε μεταφορά. Εάν η αρτηρία απασχολείται πλήρως (100%) για τέτοιες μεταφορές DMA, πόση θα είναι η συνολική παροχή της σε MBytes/s και σε Mbits/s; Στη συνέχεια, έστω (ii) ότι οι συσκευές DMA κάνουν τις αντιγραφές τους μέσω μεταφορών δύο (2) λέξεων, δηλ. 16 Bytes, κάθε φορά, οι οποίες κοστίζουν, κατά τα παραπάνω, 4 κύκλους της αρτηρίας η κάθε μεταφορά. Εάν η αρτηρία απασχολείται πάλι 100% για τέτοιες μεταφορές DMA, πόση θα είναι η συνολική παροχή της σε MBytes/s και σε Mbits/s; Ίδια ερώτηση εάν (iii) η κάθε μεταφορά στην αρτηρία αφορά burst των 4 λέξεων (32 Bytes) κάθε φορά, και (iv) εάν αφορά burst των 8 λέξεων (64 Bytes) κάθε φορά.

(γ) Εάν οι μεταφορές DMA εξυπηρετούν τους 2 δίσκους με παροχή 10 MBytes/s καθένας και το 1 δίκτυο fast ethernet με παροχή 100 Mbits/s της ερώτησης (α), και εάν οι συσκευές DMA κάνουν τις αντιγραφές τους μέσω μεταφορών bursts των 8 λέξεων (64 Bytes) κάθε φορά, τότε τι ποσοστό του χρόνου της αρτηρίας μνήμης-Ε/Ε 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), εξηγείστε 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 Ε/Ε, 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 τις μηχανές DMA, που δουλεύουν παράλληλα με τον επεξεργαστή), απασχολώντας (οι μεταφορές DMA) ένα μέρος μόνο της διαθέσιμης παροχής (throughput) της αρτηρίας μνήμης-Ε/Ε, και αφήνοντας το υπόλοιπο μέρος αυτής της παροχής διαθέσιμο για να εξυπηρετούνται οι αστοχίες της κρυφής μνήμης του επεξεργαστή.

Άσκηση 17.5: DMA και Συμβατότητα Κρυφής-Κύριας Μνήμης

Σ' ένα σύστημα όπου γίνονται μεταφορές DMA πρέπει να λυθεί το πρόβλημα της συμβατότητας κρυφής και κύριας μνήμης (πρόβλημα **Cache Coherence**). Δείξτε ποιό είναι το πρόβλημα αυτό, κάνοντας τα εξής. Σχεδιάστε (i) τον επεξεργαστή με την κρυφή του μνήμη, η οποία συνδέεται στην αρτηρία (λεωφόρο - bus) μνήμης-Ε/Ε, (ii) την κύρια μνήμη, συνδεδεμένη στην ίδια αρτηρία, και (iii) μία συσκευή Ε/Ε με μηχανισμό DMA, συνδεδεμένη στην ίδια αρτηρία.

(α) Θεωρήστε την περιφερειακή συσκευή σαν συσκευή εισόδου, και θεωρήστε ότι αυτή μεταφέρει μέσω DMA νέα δεδομένα εισόδου σε κάποια περιοχή διευθύνσεων στην κύρια μνήμη. Μετά τη λήξη της μεταφοράς, το πρόγραμμα που τρέχει στον επεξεργαστή θέλει να διαβάσει (μέσω load) τα νέα δεδομένα εισόδου από την περιοχή διευθύνσεων στην κύρια μνήμη όπου αυτά έχουν τοποθετηθεί από το DMA. Σε ποια περίπτωση θα διαβάσει τα σωστά νέα δεδομένα, και σε ποια περίπτωση θα διαβάσει λανθασμένες παλαιές τιμές;

(β) Θεωρήστε την περιφερειακή συσκευή σαν συσκευή εξόδου, και θεωρήστε ότι το πρόγραμμα που τρέχει στον επεξεργαστή παράγει μερικά νέα δεδομένα τα οποία γράφει (μέσω store) σε ορισμένη περιοχή διευθύνσεων μνήμης, και τα οποία στη συνέχεια θέλει να στείλει στην περιφερειακή συσκευή. Γιά το σκοπό αυτό, το λειτουργικό σύστημα ξεκινάει μία μεταφορά DMA από την παραπάνω περιοχή διευθύνσεων κύριας μνήμης προς τη συσκευή εξόδου. Έστω ότι η κρυφή μνήμη του επεξεργαστή είναι τύπου **write through**, δηλαδή, ως γνωστόν, κάθε τι που γράφει ο επεξεργαστής σε αυτήν, αυτή το γράφει αμέσως και στην κύρια μνήμη. Υπ' αυτές τις συνθήκες, υπάρχει περίπτωση να φτάσουν λάθος (παλαιά) δεδομένα στη συσκευη εξόδου; Γιατί όχι;

(γ) Έστω τώρα ότι στο σύστημα (β) η κρυφή μνήμη είναι τύπου **write back**, δηλαδή δεν γράφει αμέσως στην κύρια μνήμη κάθε αλλαγή τιμής (εγγραφή νέας τιμής) που κάνει ο επεξεργαστής, αλλά το γράφει αργότερα, όταν το block όπου έγινε η αλλαγή πρέπει να αντικατασταθεί στην κρυφή μνήμη από άλλο block. Υπ' αυτές τις συνθήκες, σε ποια περίπτωση θα καταλήξουν τα σωστά νέα δεδομένα στη συσκευή εξόδου, και σε ποια περίπτωση θα καταλήξουν εκεί λανθασμένες παλαιές τιμές;

Λύσεις στο πρόβλημα της συμβατότητας κρυφής και κύριας μνήμης υπάρχουν "μεσοβέζικες", με εκδίωξη (flush) σελίδων από την κρυφή μνήμη (δύσκολο ή χρονοβώρο) ή με χρήση σελίδων που η κρυφή μνήμη αναγνωρίζει και δεν κρατά (non-cacheable pages) (μειώνει την επίδοση του επεξεργαστή), ή "ριζικές", με χρήση ενός πρωτόκολλου συμβατότητας κρυφών μνημών σαν αυτά που χρησιμοποιούν οι πολυεπεξεργαστές κοινόχρηστης μνήμης (shared-memory multiprocessors).

Παραδώστε ηλεκτρονικά τις απαντήσεις σας, σε ένα αρχείο "**`ask17.pdf`**" (PDF format).

[**Up** - Table of Contents] [printer version - **PDF**]

Up to the Home Page of CS-225