Σειρά Ασκήσεων 16: Εικονική Μνήμη (Virtual Memory)

Προθεσμία έως Παρασκευή 15 Μαίου 2009, ώρα 23:59 (βδομάδα 13)

[Up - Table of Contents] [printer version - PDF]
[Prev - 15. Cache Memories] [17. I/O and DMA - Next (Sp'06)]

Βιβλίο (Ελλ. έκδοση): διαβάστε το δεύτερο ήμισυ του κεφαλαίου 7 --από § 7.4 και πέρα-- σελ. 529-583, και ιδιαίτερα τις σελ. 529-561.

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

SRAM-DRAM: Οπως 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). Γιά 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. Ετσι, οι DRAM χρησιμοποιούνται σχεδόν πάντα γιά την κατασκευή της κύριας μνήμης (main memory) των υπολογιστών, ενώ οι SRAM χρησιμοποιούνται σχεδόν πάντα γιά 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 περίπου τόσος (ίσως λίγο χαμηλότερος) αυτός ο ρυθμός αύξησης, και είναι πολύ πιθανό να συνεχίσει έτσι, αν και κάποτε μάλλον θα πέσει. Στα μέσα της πρώτης δεκαετίας του 21ου αιώνα, η τεχνολογία των DRAM βρίσκεται γύρω στο

1 Gbit 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, ένα module με 8 chips των 256 Mbits καθένα είχε συνολική χωρητικότητα 4 Gbits = 256 MBytes, ενώ ένα αντίστοιχο module με 16 τέτοια chips προσέφερε 512 MBytes. Όταν τα chips είναι 9 αντί 8, ή 18 αντί 16, τα επιπλέον chips χρησιμοποιούνται γιά 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 64 έως 512 γραμμές επί 64 έως 512 στήλες. Γιά 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. Οταν ανάψει το σύρμα αυτό, όλα τα στοιχεία μνήμης (bits) πάνω στη γραμμή αυτή διαβάζονται, δηλαδή τοποθετούν το καθένα την τιμή του (περιεχόμενό του) στο αντίστοιχο σύρμα στήλης (bit line) που διατρέχει τη στήλη του. Ετσι, στο κάτω μέρος του block της μνήμης, στις απολήξεις των συρμάτων στήλης, εμφανίζεται το περιεχόμενο όλων των bits που είναι αποθηκευμένα στην επιλεγείσα γραμμή. Ενας 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 βγεί το τελικό bit στην έξοδο, διαρκεί αρκετό χρόνο (γύρω στα 60 ns γιά τις σημερινές DRAM).

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 2 με 5 ns σε καθένα από τα σύρματα δεδομένων (data) γιά τις σημερινές DRAM). Εκμεταλλευόμενοι τη δυνατότητα αυτή, πετυχαίνουμε να προσπελαύνουμε 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.

Διαφύλλωση (Interleaving): Η άλλη τεχνική που χρησιμοποιείται γιά την αύξηση της παροχής (throughput) μιάς μνήμης --κυρίως γιά προσπελάσεις σε τυχαίες διευθύνσεις και όχι τόσο γιά 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).

16.2 Εικονική Μνήμη, Πίνακες Μετάφρασης, Προστασία Μνήμης:

Η εικονική μνήμη χρησιμοποιείται γιά τρείς κυρίως σκοπούς:

i. Προστασία μεταξύ των πολλαπλών διεργασιών (processes) που τρέχουν.
ii. Ανεξαρτησία διευθύνσεων μεταξύ των διεργασιών αυτών.
iii. Δυνατότητα η κάθε διεργασία να "βλέπει" χώρο μνήμης μεγαλύτερο από το κομμάτι της φυσικής μνήμης που όντως της διατίθεται.

Ο βασικός τρόπος λειτουργίας της εικονικής μνήμης είναι ο εξής. Κάθε διεύθυνση μνήμης που γεννά ο επεξεργαστής --δηλαδή το πρόγραμμα που τρέχει-- θεωρείται ως "**εικονική διεύθυνση**", και **μεταφράζεται** σε μιάν άλλη, "**φυσική διεύθυνση**", προτού δοθεί στη μνήμη γιά να επιλεγεί η λέξη την οποία τελικά θα προσπελάσει το πρόγραμμα. Η μετάφραση αυτή φροντίζει:

i. Να ελέγχει ότι η διεργασία που τρέχει έχει δικαίωμα να κάνει την προσπέλαση που ζητά (ανάγνωση/εγγραφή/εκτέλεση) στη διεύθυνση που ζητά.
ii. Να μεταφράζει τις εικονικές διευθύνσεις της κάθε διεργασίας σε διαφορετικές φυσικές διευθύνσεις γιά την κάθε διεργασία, εκτός των περιπτώσεων που θέλουμε οι διεργασίες να επικοινωνούν μεταξύ τους μέσω κοινόχρηστης μνήμης (shared memory).
iii. Να μεταφράζει τις πιό συχνά (επί του παρόντος) χρησιμοποιούμενες εικονικές διευθύνσεις στις φυσικές διευθύνσεις όπου αυτές "χωράνε", ενώ όσες δεν χωράνε στην υπάρχουσα φυσική μνήμη προκαλούν σφάλμα σελίδας (page fault - § 13.1), ώστε να φροντίσει το λειτουργικό σύστημα να τις φέρει (συνήθως από το δίσκο).

Η μετάφραση διευθύνσεων γίνεται απεικονίζοντας ολόκληρες "**σελίδες**" *(pages)* εικονικής μνήμης σε ολόκληρες φυσικές σελίδες. Το μέγεθος της σελίδας είναι αρκετά KBytes σήμερα, και η τάση είναι να μεγαλώνει με τα χρόνια. Γιά να γίνεται η μετάφραση γρήγορα, χρησιμοποιείται συνήθως ένας μικρός κατάλογος ζευγών εικονικής-φυσικής σελίδας γιά τις πιό συχνά χρησιμοποιούμενες σελίδες --ο "**TLB**" *(Translation Lookaside Buffer)*-- οργανωμένος σαν μιά μικρή κρυφή μνήμη, συνήθως πλήρως προσεταιριστική. Οταν μιάν εικονική σελίδα δεν την βρίσκουμε στον TLB, τότε την αναζητάμε στους *Πίνακες Μετάφρασης*, που βρίσκονται στη μνήμη.

Θεωρήστε το εξής μικρό (εξωπραγματικό σήμερα) σύστημα εικονικής μνήμης, σαν απλό παράδειγμα.

- Οι εικονικές διευθύνσεις έχουν μέγεθος 20 bits (δηλ. είναι πενταψήφιες στο δεκαεξαδικό σύστημα), άρα ο χώρος εικονικών διευθύνσεων είναι 1 MByte ανά διεργασία.
- Μέγεθος σελίδας = 4 KBytes, άρα τα 12 LS bits κάθε διεύθυνσης (3 LS δεκαεξαδικά ψηφία) επιλέγουν 1 byte μέσα στη σελίδα, ενώ τα υπόλοιπα MS bits υποδηλώνουν γιά ποιά σελίδα μιλάμε. Ετσι, η κάθε διεργασία έχει 256 εικονικές σελίδες (1 MByte / 4 KBytes = 256).
- Η φυσική μνήμη είναι 64 KBytes, άρα οι φυσικές διευθύνσεις αποτελούνται από 16 bits (4 δεκαεξαδικά ψηφία), επομένως υπάρχουν 16 φυσικές σελίδες (16 KBytes / 4 KBytes = 16).

Τότε, η μετάφραση μιάς εικονικής διεύθυνσης, π.χ. της FE210, στην αντίστοιχη φυσική γίνεται ως εξής:

- Χωρίζουμε την εικονική διεύθυνση στα 12 LS bits, που υποδηλώνουν το byte μέσα στη σελίδα (εδώ: 210), και στα 8 MS bits, που ορίζουν την εικονική σελίδα (εδώ: FE).
- Τα 12 LS bits (210) δεν χρειάζονται μετάφραση, αφού απεικονίζουμε ολόκληρες εικονικές σελίδες σε ολόκληρες φυσικές σελίδες, και όλες οι σελίδες είναι ευθυγραμμισμένες στα φυσικά όριά τους.
- Τα 8 MS bits, δηλαδή ο αριθμός εικονικής σελίδας (FE), χρησιμοποιούνται σαν index στον πίνακα μετάφρασης της διεργασίας (process) που τρέχει αυτή τη στιγμή. Ο πίνακας αυτός έχει μέγεθος 256 θέσεις --όσες και οι εικονικές σελίδες ανά διεργασία. Υπάρχει **χωριστός** πίνακας μετάφρασης **γιά κάθε διεργασία**, έτσι ώστε η κάθε διεργασία να μπορεί να έχει τις δικές της, διαφορετικές, φυσικές σελίδες, παρά το γεγονός ότι χρησιμοποιεί ίδιες εικονικές διευθύνσεις με όλες τις άλλες διεργασίες.
- Στη θέση FE του πίνακα μετάφρασης, όπου μας οδήγησαν τα 8 MS bits της εικονικής διεύθυνσης, υπάρχουν πληροφορίες --όπως θα πούμε παρακάτω-- γιά να ελέγξουμε αν η εικονική σελίδα FE που θέλουμε υπάρχει στη φυσική μνήμη, και αν έχουμε δικαίωμα να την προσπελάσουμε όπως ζητά η τρέχουσα διεργασία.
- Στη ίδια θέση FE του πίνακα μετάφρασης βρίσκουμε τον αριθμό της φυσικής σελίδας όπου βρίσκεται αυτή τη στιγμή η εικονική σελίδα FE. Στο παράδειγμά μας, υπάρχουν 16 φυσικές σελίδες, άρα ο αριθμός φυσικής σελίδας έχει 4 bits. Έστω ότι βρήκαμε τον αριθμό A σαν αριθμό φυσικής σελίδας. Σε αυτόν κολάμε και τα 12 αμετάφραστα LS bits της εικονικής διεύθυνσης (210), και φτιάχνουμε έτσι τα 16 bits της φυσικής διεύθυνσης: A210, στο παράδειγμά μας.

Διαχωρισμός και Προστασία Διεργασιών: το hardware του επεξεργαστή βρίσκει τον πίνακα μετάφρασης της τρέχουσας διεργασίας από τη (φυσική) διεύθυνση βάσης του πίνακα αυτού, που είναι γραμμένη (από το λειτουργικό σύστημα) σ' έναν ειδικό καταχωρητή του συστήματος διαχείρισης μνήμης --όχι

στο κανονικό register file. Όταν ο επεξεργαστής τρέχει σε "**user mode**", δεν επιτρέπεται να γράψει αυτόν τον καταχωρητή, ούτως ώστε να μην μπορεί να υποκριθεί ότι είναι άλλη διεργασία, δηλαδή να μην μπορεί να αποκτήσει πρόσβαση στη μνήμη άλλων διεργασιών.

Προστασία Λειτουργικού Συστήματος: Ο παραπάνω ειδικός καταχωρητής που καθορίζει τον τρέχοντα πίνακα μετάφρασης --δηλαδή την τρέχουσα διεργασία-- είναι προσπελάσιμος από τον επεξεργαστή μόνον όταν ο επεξεργαστής βρίσκεται σε "**kernel mode**", δηλαδή τρέχει το λειτουργικό σύστημα. Κάθε εξαίρεση (exception) --περιλαμβανόμενου και του καλέσματος συστήματος (system call)-- αποθηκεύει την παλιά κατάσταση (user/kernel) στην οποία έτρεχε ο επεξεργαστής, και φέρνει τον επεξεργαστή σε kernel mode. Έτσι, ο trap (exception) handler εκτελείται πάντα σε kernel mode, ενώ ο μόνος τρόπος γιά ένα χρήστη να φέρει τον επεξεργαστή σε kernel mode είναι να προκαλέσει εξαίρεση, εκτελόντας μιάν εντολή system call --κάτι σαν παράνομη εντολή που προκαλεί εξαίρεση, αλλά που το λειτουργικό σύστημα ξέρει ότι προορίζεται σαν system call και όχι σαν απλή παράνομη εντολή λόγω προγραμματιστικού σφάλματος. Το κάλεσμα συστήματος είναι επίτηδες φτιαγμένο να συμπεριφέρεται σαν εξαίρεση (exception), και όχι σαν απλό κάλεσμα διαδικασίας (εντολή jal), ούτως ώστε η είσοδος στο λειτουργικό σύστημα --που πρέπει να γίνει σε kernel mode-- να γίνεται μόνο στην προκαθορισμένη διεύθυνση του trap handler, και όχι σε οιαδήποτε άλλη αυθαίρετη διεύθυνση θα μπορούσε να ζητήσει ένας κακόβουλος χρήστης προκειμένου να παρακάμψει το μέρος εκείνο του λειτουργικού συστήματος που κάνει τους ελέγχους του εάν ο χρήστης έχει δικαίωμα να ζητήσει αυτό που ζητά.

Παρούσες/Απούσες Σελίδες και Προστασία Σελίδων: Κάθε θέση του πίνακα μετάφρασης περιέχει:

- Το **"valid bit**", που υποδεικνύει αν η εικονική σελίδα στην οποία αναφερόμαστε είναι παρούσα ή απούσα από τη φυσική μνήμη. Ενδεχόμενη απουσία της εικονικής σελίδας από τη φυσική μνήμη μπορεί να οφείλεται είτε στο ότι η εικονική αυτή σελίδα είναι *παράνομη* (δεν χρησιμοποιείται, δηλαδή το πρόγραμμα δεν είχε εντολές ή δεδομένα εκεί, ούτε ζήτησε να βάλει κάτι μέσω malloc/sbreak), είτε στο ότι είναι νόμιμη μεν αλλά αυτή τη στιγμή βρίσκεται *στο δίσκο* και όχι στη μνήμη.
- Τον **αριθμό της φυσικής σελίδας** (4 bits στο εδώ παράδειγμά μας) όπου βρίσκεται αυτή τη στιγμή η εικονική σελίδα, όταν αυτή είναι παρούσα στη φυσική μνήμη.
- Τα "**bits προστασίας**" (π.χ. 3 bits: "rwx"), που δηλώνουν τι είδους προσπελάσεις επιτρέπεται να κάνει η παρούσα διεργασία στις λέξεις αυτής της σελίδας.
- Το "**dirty bit**", που δείχνει αν ο επεξεργαστής άλλαξε ή όχι τα περιεχόμενα

αυτής της σελίδας από τότε που την διαβάσαμε από το δίσκο.
- Το "**reference bit**", που το χρησιμοποιεί το λειτουργικό σύστημα γιά να προσεγγίσει τον αλγόριθμο αντικατάστασης "LRU": σε κάθε προσπέλαση στη σελίδα, ο επεξεργαστής θέτει αυτό το bit, ενώ το λειτουργικό σύστημα περιοδικά διαβάζει αυτά τα bits γιά να δεί ποιές σελίδες χρησιμοποιήθηκαν πρόσφατα, και στη συνέχεια τα μηδενίζει.

Άσκηση 16.3: Μονοεπίπεδος Πίνακας Μετάφρασης

(α) Γιά το παραπάνω μικρό (εξωπραγματικό σήμερα) παράδειγμα εικονικής μνήμης, κάντε ένα σχηματικό διάγραμμα που να δείχνει τον καταχωρητή που περιέχει τον pointer στον πίνακα μετάφρασης της παρούσας διεργασίας, τον πίνακα μετάφρασης, την εικονική διεύθυνση (20 bits) που γεννά ο επεξεργαστής, τα πεδία από τα οποία αυτή αποτελείται, από που προέρχεται το index στον πίνακα μετάφρασης, τι διαβάζουμε από τη θέση εκείνη του πίνακα, και πώς συνθέτουμε τη φυσική διεύθυνση (16 bits).

(β) Έστω ότι, στο παραπάνω απλό παράδειγμά μας, η διεργασία μας έχει τις εξής σελίδες:

- Εικονική σελίδα 00: παράνομη (περιέχει τον NULL pointer).
- Εικονική σελίδα 01: περιέχει κώδικα (r-x), και βρίσκεται στη φυσική σελίδα 7.
- Εικονική σελίδα 02: περιέχει κώδικα (r-x), και βρίσκεται στη φυσική σελίδα C.
- Εικονική σελίδα 03: περιέχει στατικά δεδομένα (rw-), και βρίσκεται στη φυσική σελίδα 0, dirty.
- Εικονικές σελίδες 04 έως και 09: απούσες από τη φυσική μνήμη.
- Εικονικές σελίδες 0A έως και 9F: παράνομες (unallocated).
- Εικονική σελίδα A0: περιέχει δυναμικά δεδομένα (r--), και βρίσκεται στη φυσική σελίδα D.
- Εικονική σελίδα A1: περιέχει δυναμικά δεδομένα (rw-), και βρίσκεται στη φυσική σελίδα E, dirty.
- Εικονικές σελίδες A2 έως και A5: απούσες από τη φυσική μνήμη.
- Εικονικές σελίδες A6 έως και FD: παράνομες (unallocated).
- Εικονική σελίδα FE: περιέχει δεδομένα στοίβας (rw-), και βρίσκεται στη φυσική σελίδα A, dirty.
- Εικονική σελίδα FF: περιέχει δεδομένα στοίβας (rw-), και βρίσκεται στη φυσική σελίδα 1, clean.

Δείξτε τα περιεχόμενα του πίνακα μετάφρασης της διεργασίας μας, χωρίς τα reference bits αλλά με όλα τα άλλα πεδία του (256 γραμμές επί 4 πεδία ανά γραμμή --επιτρέπεται η χρήση αποσιωποιητικών...).

(γ) Ποιές από τις παρακάτω προσπελάσεις στις εικονικές διευθύνσεις που δίδονται προκαλούν σφάλμα σελίδας; Οι υπόλοιπες, σε ποιά φυσική διεύθυνση μεταφράζονται;

02038 (fetch), 03FF4 (read), A001C (write), 0192C (fetch), 00000 (read), 92FC0 (read), FE5D8 (write), 03FF4 (fetch), A1FFC (read), 008F4 (write), A2000 (read), 01E40 (write).

Άσκηση 16.4: Πολυεπίπεδοι Πίνακες Μετάφρασης - Οικονομία Χώρου

Οι περισσότερες διεργασίες χρησιμοποιούν σχετικά λίγες από τις εικονικές τους σελίδες, και αυτές συνοθηλευμένες (clustered) σε μερικές "γειτονιές", όπως στην παραπάνω άσκηση 16.3. Εκμεταλλευόμενοι αυτή την ιδιότητα, μπορούμε να μειώσουμε το χώρο μνήμης που καταλαμβάνει ο πίνακας μετάφρασης της κάθε διεργασίας, σπάζοντας τον σε μερικούς μικρότερους πίνακες, οργανωμένους σαν μιά πολυ-επίπεδη ιεραρχία.

Θεωρήστε ότι στο σύστημα μνήμης της άσκησης 16.3 αλλάζουμε τον μοναδικό (μονοεπίπεδο) πίνακα μετάφρασης ανά διεργασία σε διεπίπεδους πίνακες, ώς εξής. Κάθε διεργασία έχει έναν πίνακα πρώτου επιπέδου, μεγέθους 16 θέσεων. Τον πίνακα αυτόν τον βρίσκουμε μέσω του γνωστού pointer που περιέχεται στον ειδικό καταχωρητή που αναφέραμε παραπάνω. Χρησιμοποιούμε τα 4 MS bits της εικονικής διεύθυνσης γιά να επιλέξουμε μία από τις 16 θέσεις αυτού του πίνακα. Κάθε συνδυασμός των 4 αυτών bits, επομένως και κάθε θέση αυτού του πίνακα, αντιστοιχεί σε 16 εικονικές σελίδες. Εαν καμία από αυτές τις 16 σελίδες δεν υπάρχει στη φυσική μνήμη, τότε σημειώνουμε τη θέση αυτή του πίνακα πρώτου επιπέδου σαν άκυρη (valid bit = 0). Αλλοιώς, η θέση αυτή του πίνακα πρώτου επιπέδου περιέχει έναν pointer σε ένα πίνακα μετάφρασης δευτέρου επιπέδου. Εάν η εικονική μας διεύθυνση μας οδήγησε σε τέτοια θέση στον πίνακα πρώτου επιπέδου, τότε χρησιμοποιούμε τα επόμενα 4 bits της εικονικής διεύθυνσης σαν index στον πίνακα δευτέρου επιπέδου όπου μας οδήγησε ο πίνακας πρώτου επιπέδου. Εκεί, στον πίνακα δευτέρου επιπέδου, βρίσκουμε τα τελικά στοιχεία γιά τη σελίδα που ζητάμε.

(αβγ) Κάντε ένα διάγραμμα ανάλογο προς αυτό της άσκησης 16.3(α) γιά το διεπίπεδο σύστημα μετάφρασης αυτής της άσκησης. Στο ίδιο διάγραμμα, δείξτε όλους τους πίνακες δευτέρου επιπέδου που θα υπάρχουν γιά τις σελίδες της άσκησης 16.3(β). Επίσης δείξτε, όλα τα περιεχόμενα όλων των πινάκων μετάφρασης, πρώτου και δευτέρου επιπέδου. Βεβαιωθείτε (χωρίς να δώσετε γραπτά την απάντησή σας) ότι το σύστημα αυτό μεταφράζει τις διευθύνσεις της άσκησης 16.3(γ) το ίδιο όπως και το μονοεπίπεδο σύστημα της άσκησης εκείνης.

(δ) Πόσες θέσεις μνήμης καταλαμβάνουν όλοι οι πίνακες μετάφρασης του παρόντος διεπίπεδου συστήματος γιά τη διεργασία μας και γιά τις σελίδες (β); Σε σχέση με το μονοεπίπεδο σύστημα της άσκησης 16.3(β) υπάρχει οικονομία στο χώρο μνήμης που καταλαμβάνεται;

Άσκηση 16.5: Πολυεπίπεδοι Πίνακες Μετάφρασης σε ένα Ρεαλιστικό Σύστημα

Επαναλάβετε την άσκηση 16.4(α) --που ήταν σαν την 16.3(α) (σχηματικό διάγραμμα πινάκων μετάφρασης)-- αυτή τη φορά γιά ένα ρεαλιστικό, σημερινό, σύστημα εικονικής μνήμης:

- Οι εικονικές διευθύνσεις έχουν μέγεθος 64 bits. Όμως, γιά πρακτικούς λόγους, μονο τα 46 LS από αυτά χρησιμοποιούνται (χώρος εικονικών διευθύνσεων 16 TeraBytes ανά διεργασία...). Γιά να μην μπουν οι χρήστες στο πειρασμό να "εκμεταλλευτούν" γιά άλλο σκοπό τα 18 MS bits του κάθε pointer που περισσεύουν --χάνοντας έτσι το portability των προγραμμάτων τους σε μελλοντικές γενιές συστημάτων με περισσότερα υλοποιημένα bits εικονικής διεύθυνσης-- το σύστημά μας ελέγχει κάθε προσπέλαση μνήμης γιά να σιγουρευτεί ότι αυτά τα 18 MS bits της διεύθυνσης ισούνται όλα με το MS bit των 46 LS bits, δηλαδή αποτελούν επέκταση προσήμου (sign extension) του υλοποιημένου μέρους της εικονικής διεύθυνσης (έτσι, μπορεί η στοίβα να συνεχίσει να τοποθετείται στο άνω άκρο του εικονικού χώρου μνήμης (διευθύνσεις FFFFFF...).
- Μέγεθος σελίδας 64 KBytes. Επομένως τα 16 LS bits της εικονικής διεύθυνσης δεν χρειάζονται μετάφραση.
- Τρία επίπεδα πινάκων μετάφρασης ανά διεργασία.
- Κάθε πίνακας μετάφρασης έχει μέγεθος 1024 θέσεις, άρα "κοιτάζει" 10 από τα bits του αριθμού εικονικής σελίδας.

Άσκηση 16.6: TLB, Process ID, και Κοινόχρηστες Σελίδες

Όπως είπαμε και παραπάνω, γιά να γίνεται η μετάφραση γρήγορα, χρησιμοποιείται συνήθως ένας μικρός κατάλογος ζευγών εικονικής-φυσικής σελίδας γιά τις πιό συχνά χρησιμοποιούμενες σελίδες, ο "**TLB**" (Translation Lookaside Buffer), οργανωμένος σαν μιά μικρή κρυφή μνήμη, συνήθως πλήρως προσεταιριστική.

Προκειμένου να μην αναγκαζόμαστε να ακυρώνουμε τα περιεχόμενα του TLB σε κάθε αλλαγή της διεργασίας που τρέχει (context swap), θέλουμε να μπορούμε να έχουμε μέσα στο TLB, ταυτόχρονα, ζευγάρια εικονικής-φυσικής σελίδας πολλών διαφορετικών διεργασιών. Αυτό όμως απαιτεί να μπορούμε να τα ξεχωρίζουμε μεταξύ τους, αφού την κάθε ορισμένη εικονική διεύθυνση ενδέχεται να την χρησιμοποιούν πολλές διεργασίες αλλά γιά διαφορετική πληροφορία και κατά

διαφορετικό τρόπο η κάθεμία. Γιά να γίνεται ο διαχωρισμός αυτός, καταγράφουμε τον αριθμό διεργασίας ("**PID**", Process Identifier) μαζί με τον αριθμό εικονικής σελίδας αυτής της διεργασίας σε κάθε θέση (ζευγάρι εικονικής-φυσικής σελίδας) του TLB.

(α) Θεωρήστε την εικονική μνήμη της άσκησης 16.3, και θεωρήστε ότι το PID έχει μέγεθος 8 bits (μέχρι 256 ταυτόχρονες διεργασίες). Θεωρήστε ένα TLB μεγέθους 16 θέσεων, με πλήρως προσεταιριστική τοποθέτηση ζευγών (οιοδήποτε ζεύγος μετάφρασης μπορεί να μπεί οπουδήποτε στο TLB). Ποιά πεδία πρέπει να έχει η κάθε θέση αυτού του TLB, και τι μεγέθους το καθένα;

(β) Δώστε ένα αριθμητικό παράδειγμα του πλήρους περιεχομένου του TLB όταν αυτό περιέχει ζευγάρια μετάφρασης γιά τις εξής σελίδες:

- Την εικονική σελίδα 01 της διεργασίας 3B (περιέχει κώδικα), και την εικονική σελίδα 01 της διεργασίας B4 (περιέχει data r/w), οι οποίες αναφέρονται σε εντελώς διαφορετικές, ανεξάρτητες φυσικές σελίδες.
- Η διεργασία 3B και η διεργασία 3C αποτελούν διαφορετικές ενεργοποιήσεις του ιδίου προγράμματος, π.χ. ο ίδιος web browser τρεγμένος από δύο διαφορετικούς χρήστες. Γιά οικονομία (φυσικής) μνήμης, το λειτουργικό σύστημα κρατά μόνο ένα αντίτυπο του κώδικα αυτής της εφαρμογής στη μνήμη, και κάνει όλες τις διεργασίες που το τρέχουν να "βλέπουν" αυτό το μοναδικό κοινόχρηστο αντίτυπο στη δική της εικονική μνήμη η κάθε μία. Βάλτε λοιπόν στο TLB και την εικονική σελίδα 01 της διεργασίας 3C, που περιέχει τον ίδιο κώδικά με τη σελίδα 01 της 3B.
- Βάλτε την εικονική σελίδα FF της διεργασίας 3B, και την εικονική σελίδα FF της διεργασίας 3C, που περιέχουν δεδομένα στοίβας. Οπως είπαμε, αυτές οι δύο διεργασίες αποτελούν διαφορετικές ενεργοποιήσεις του ιδίου προγράμματος. Παρά το γεγονός ότι ο κώδικας των δύο αυτών διεργασιών είναι κοινός, τα δεδομένα τους όμως είναι διαφορετικά, αφού π.χ. άλλα κάνει ο ένας χρήστης με τον web browser του, και άλλα ο άλλος.
- Οι διεργασίες D2 και D3 συνεργάζονται μεταξύ τους, και επικοινωνούν μέσω κοινόχρηστης μνήμης: Η διεργασία D2 παράγει δεδομένα προς επεξεργασία, και τα γράφει στην εικονική σελιδα της A0 (προστασία -w-). Η διεργασία D3 καταναλώνει τα δεδομένα που παράγει η D2, και τα επεξεργάζεται. Η D3 διαβάζει τα δεδομένα αυτά από την εικονική σελίδα της C0 (προστασία r--), η οποία όμως, μέσω του συστήματος εικονικής μνήμης, αντιστοιχεί στην ίδια φυσική σελίδα στην οποία αντιστοιχεί και η A0 της D2, ώστε να επιτυγχάνεται η συνεργασία παραγωγού-καταναλωτή των δύο διεργασιών. Βάλτε στο TLB σας και αυτές τις δύο εικονικές σελίδες.

(γ) Οι διεργασίες 3B και 3C, παραπάνω, είναι προστατευμένες η μία από την άλλη; Μπορεί η μία να διαβάσει τα δεδομένα της άλλης (κλέβοντας έτσι, π.χ., ο

ένας χρήστης τις εμπιστευτικές πληροφορίες που ο άλλος διαβάζει μέσω διαδικτύου); Μπορεί η μία να αλλοιώσει (γράψει) τα δεδομένα της άλλης (παραπλανόντας έτσι, π.χ., ο ένας χρήστης τον άλλον); Μπορεί η μία να καταστρέψει (γράψει) τον κώδικα της άλλης ("κολλώντας" έτσι, π.χ., ο ένας χρήστης τον άλλον); Πώς εξασφαλίζουμε την επιθυμητή προστασία και ανεξαρτησία μεταξύ αυτών των δύο διεργασιών, ενώ ταυτόχρονα κάνουμε και οικονομία μνήμης κρατώντας ένα μόνο φυσικό αντίτυπο του κώδικα που αυτές τρέχουν;

Τρόπος Παράδοσης: Παραδώστε όλες τις απαντήσεις σας **σε χαρτί**, στο μάθημα. (Εάν γράψετε την απάντηση σε υπολογιστή, παρακαλείστε να την τυπώσετε και να παραδώσετε μόνο χαρτί, γιά λόγους ομοιομορφίας και διευκόλυνσης της διόρθωσης).

[Up - Table of Contents]
[Prev - 15. Cache Memories]
[printer version - PDF]
[17. I/O and DMA - Next (Sp'06)]

Up to the Home Page of CS-225