Επικοινωνία του Επεξεργαστή με Περιφερειακές Συσκευές

13b (§13.4-7) – 22-24 Απριλίου 2024 – Μανόλης Κατεβαίνης

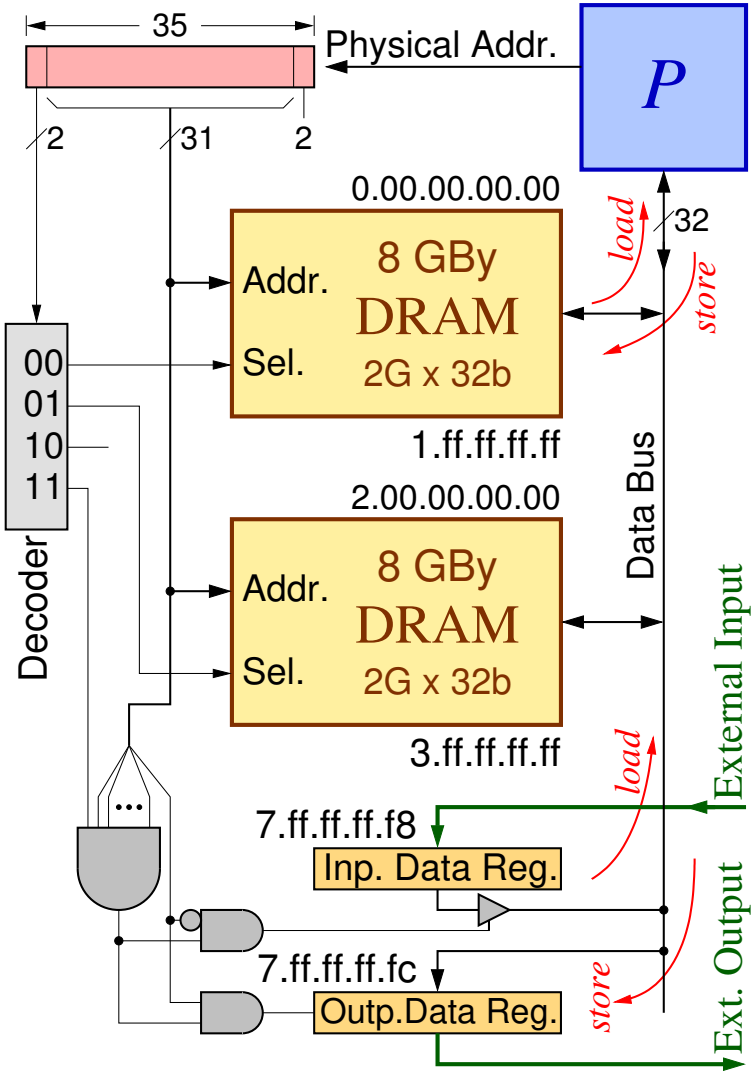
I/O Instructions versus Memory-Mapped I/O

Έχουν υπάρξει στο παρελθόν ειδικές εντολές:

- *Input*: ανάγνωση πληροφορίας από εξωτερική πηγή
- *Output*: αποστολή πληροφ. σε εξωτερ. προορισμό

Σήμερα συνήθως: *Memory-Mapped Input/Output (I/O)*
(Απεικόνιση Μνήμης των Μονάδων Εισόδου/Εξόδου (Ε/Ε)):

- *Load*: ανάγνωση όπως από μνήμη, αλλά από εξωτ. πηγή
- *Store*: όπως εγγραφή σε μνήμη, αλλά σε εξωτερ. προορ.
- Διαφοροποίηση Μονάδων Ε/Ε από κανον. μνήμη βάσει Διεύθυνσης: τμήμα του χώρου (φυσικών) διευθύνσεων αφιερωμένο σε συσκευές επικοινωνίας, αντί μνήμης

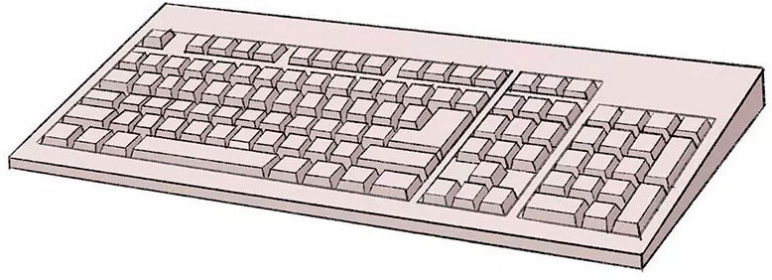


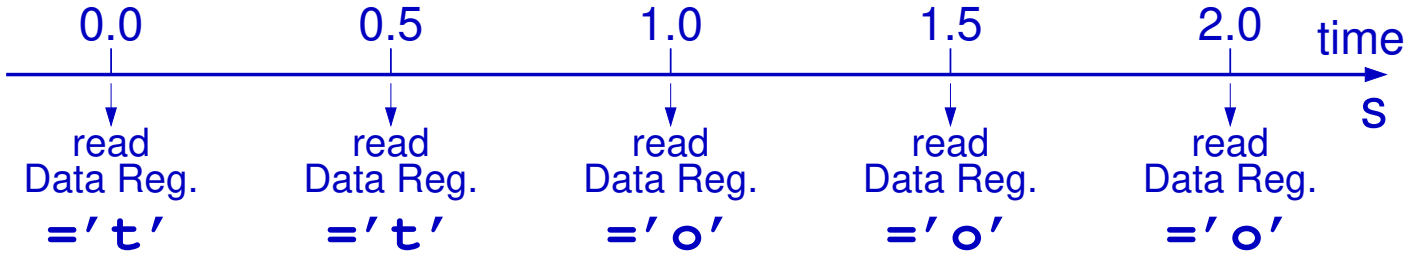
Memory-Mapped I/O

- Ορισμένες διευθύνσεις δεν είναι κανονική μνήμη, αλλά ειδικοί καταχωρητές για επικοινωνία
- *Φυσικές* Διευθύνσεις, συνήθως απεικονισμένες *μόνον* στο Λειτουργικό Σύστημα, ώστε κανείς χρήστης να μην μονοπωλεί τις συσκευές Ε/Ε
- Ειδικοί καταχωρητές:
  ένας γράφει, άλλος διαβάζει
- Αρκεί μόνον Καταχωρ. Δεδομένων για επιτυχή επικοινωνία;

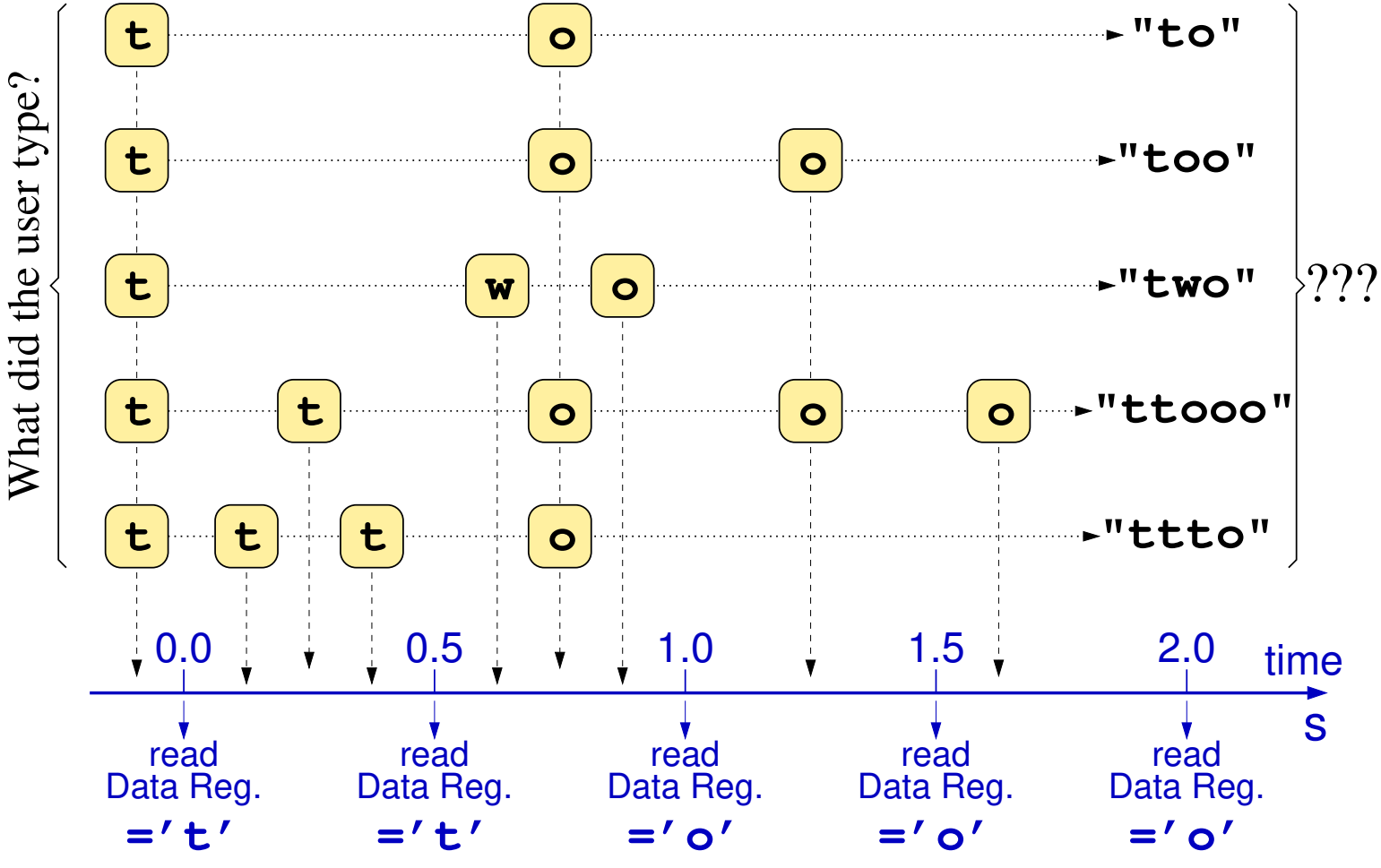
Αρκεί μόνον Data Register? Π.χ. Πληκτρολόγιο

What did the user type?



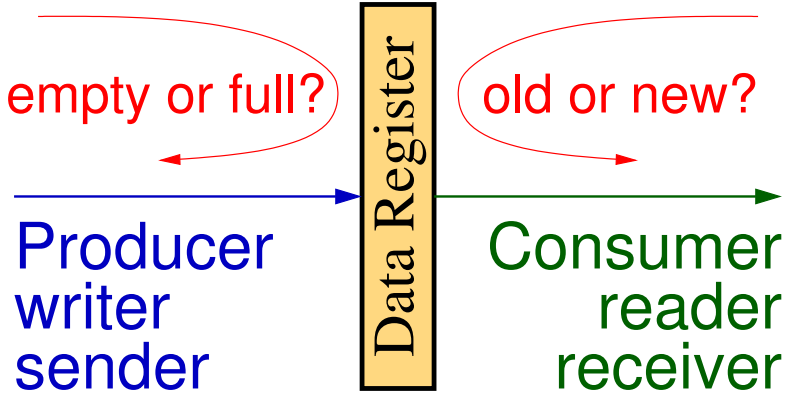


Αρκεί μόνον Data Register? Π.χ. Πληκτρολόγιο

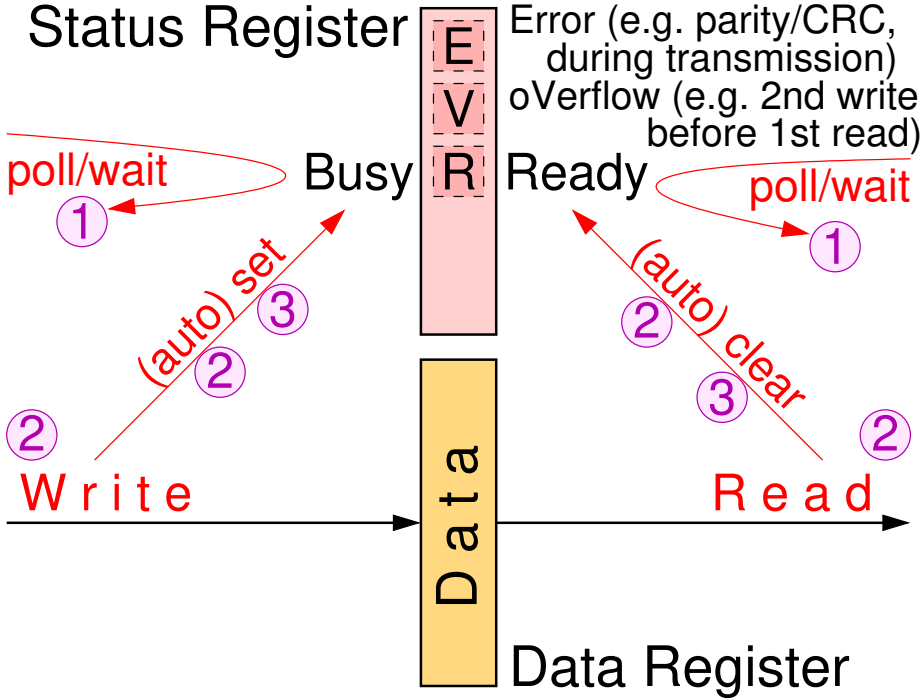


Επικοιν. Ανεξάρτητων Διεργασ. απαιτεί Συγχρονισμό

- Ανεξάρτητες διεργασίες, καθεμιά τρέχει με το ρυθμό της
  - Επεξεργαστής – Περιφερειακές Συσκευές (Ε/Ε)
  - Πολυεπεξεργαστές, πολυπύρηνοι, παράλληλα προγράμματα
- Όποτε ο ένας γράφει κάτι και ο άλλος το διαβάζει:
  - Πώς ξέρει ο καταναλωτής πότε ο παραγωγός έγραψε κάτι καινούργιο για να το διαβάσει;
  - Πώς ξέρει ο παραγωγός πότε ο καταναλωτής διάβασε το προηγούμενο για να του γράψει κάτι νέο;
- Συγχρονισμός, Έλεγχος Ροής



Status Register: Συγχρονισμός – Έλεγχος Ροής

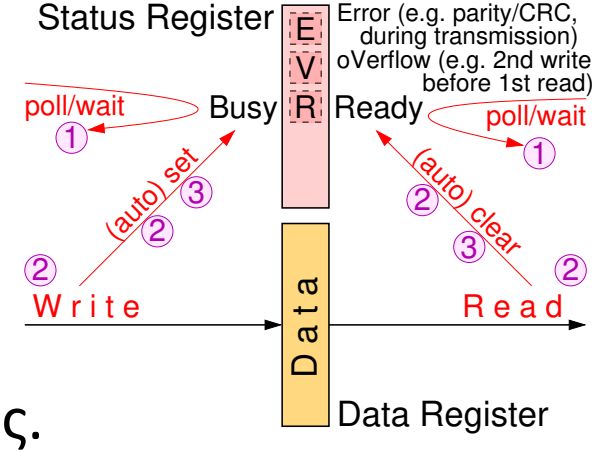


- "Control" or "Status" Register: δεύτερος καταχωρητής, επιπλέον του Data Register, σε «διπλανή» διεύθυνση μνήμης
- Ένα flag σε αυτόν:
  - Είσοδος: νέα data ήλθαν
  - Έξοδος: υπάρχει χώρος
- Άλλα flags για λάθη όπως μετάδοσης ή υπερχείλισης

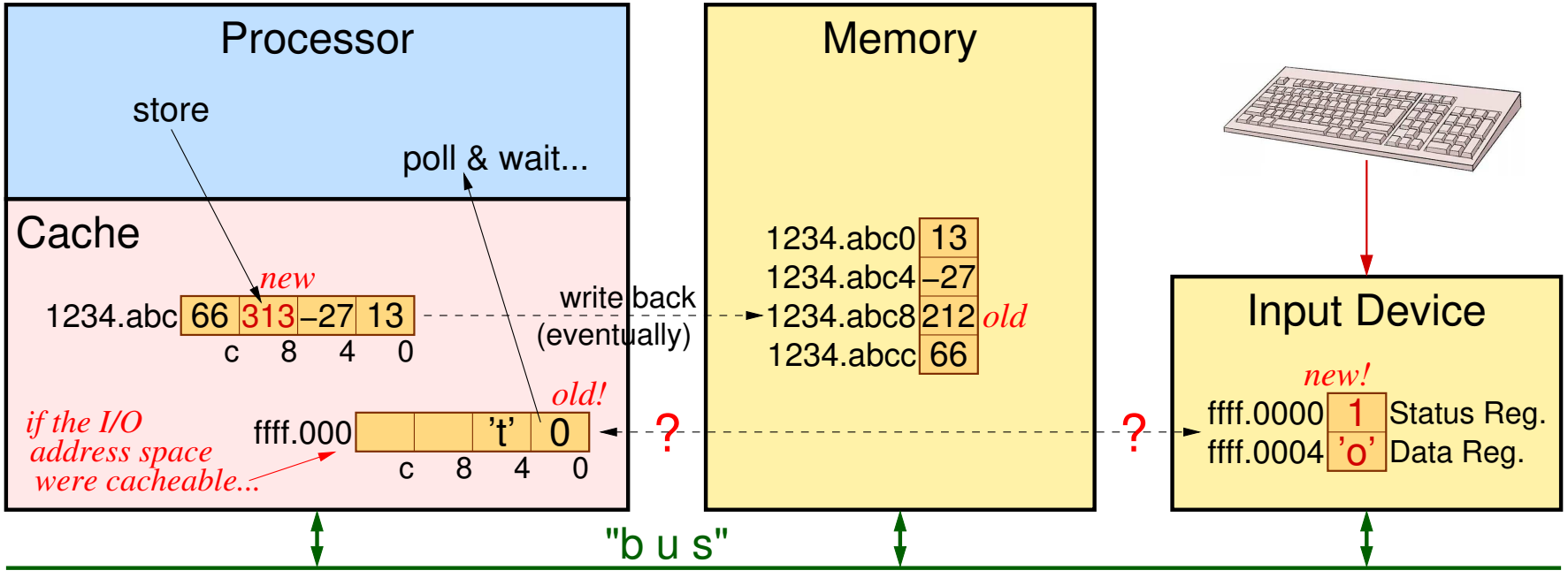
- Πρώτα περιμένουμε Ready/non-Busy, μετά διαβάζουμε/γράφουμε και ταυτόχρονα (HW) ή αμέσως μετά (SW) αλλάζουμε το status bit
- ["Off-band signaling": όταν όλοι οι συνδυασμοί των data bits είναι νόμιμοι, δηλ. όχι όπως π.χ. συμβαίνει με το NULL Byte που θεωρείται string terminator]

Polling (Δειγματοληψία), Busy-Wait και εναλλακτικές

- Polling («δειγματοληψία», ή «περιόδευση»):
  - Ο τρόπος αναμονής για νέα δεδομένα εισόδου, ή για δυνατότητα επόμενης εξόδου, μέσω επανειλημμένης ανάγνωσης του Status Register – Ready/Busy bit, μέχρις αυτό να γίνει έτοιμο
- Busy-Wait:
  - Συνεχής ανάγνωσή του, που δεν αφήνει τίποτα άλλο χρήσιμο να γίνει, μέχρις ότου το Ready bit δείξει ετοιμότητα
  - Εναλλακτικά: το διαβάζουμε περιοδικά, ενώ ενδιάμεσα κάνουμε και άλλες εργασίες. Για εξασφάλιση ότι ποτέ δεν θα αργήσουμε υπερβολικά, π.χ.: σε κάθε interrupt από το real-time clock (κάθε ~10-20 ms)

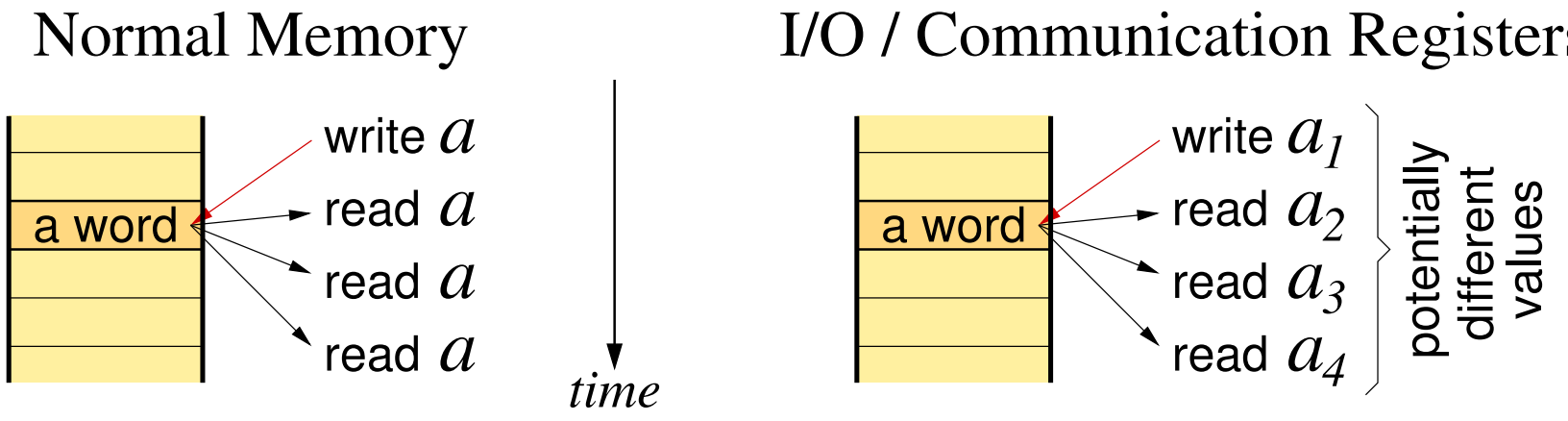


Ο χώρος διευθ. I/O πρέπει να είναι Non-Cacheable



- Όποτε θέσεις «μνήμης» προσπελάζονται από πολλαπλούς παράγοντες (I/O ή πολύεπεξεργ.), τυχόν αντίγραφα σε κρυφές μνήμες απαιτούν ειδική φροντίδα
- Εδώ: Σελίδες non-cacheable (απαγορεύεται το caching). Αλλού: Συνοχή Κρ. Μν.
- Σημειώστε: το write-through είναι ημίμετρο – μόνο αποστολή, όχι ανάγνωση

I/O/comm. Registers ≠ Normal (non-shared) Memory

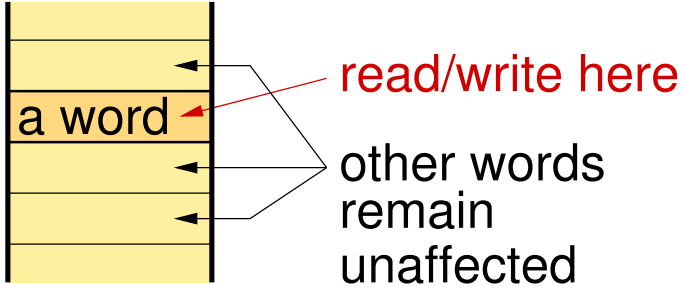


- Κάθε ανάγνωση πάντοτε επιστρέφει την πιο πρόσφατα εγγραφείσα τιμή σε αυτή τη λέξη

- Διαδοχικές αναγνώσεις μπορεί να επιστρέφουν διαφορετικές τιμές!
  - και ≠ από την τελευταία εγγραφή
  - χωρίς ενδιάμεσες εγγραφές από τον ίδιο επεξεργαστή (αλλά από άλλους επεξ./I/O)
- "**`Volatile`**" declaration in C
  - αντίγραφα σε καταχωρητή μπορεί λάθος

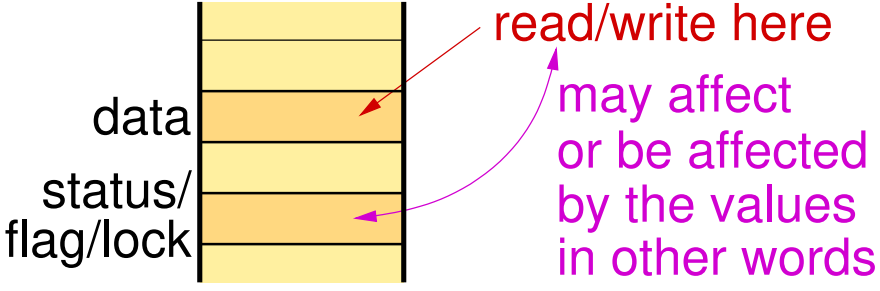
Αλληλουχία προσπελάσεων σημαντική, *και* σε ≠ διευθ.

Normal Memory



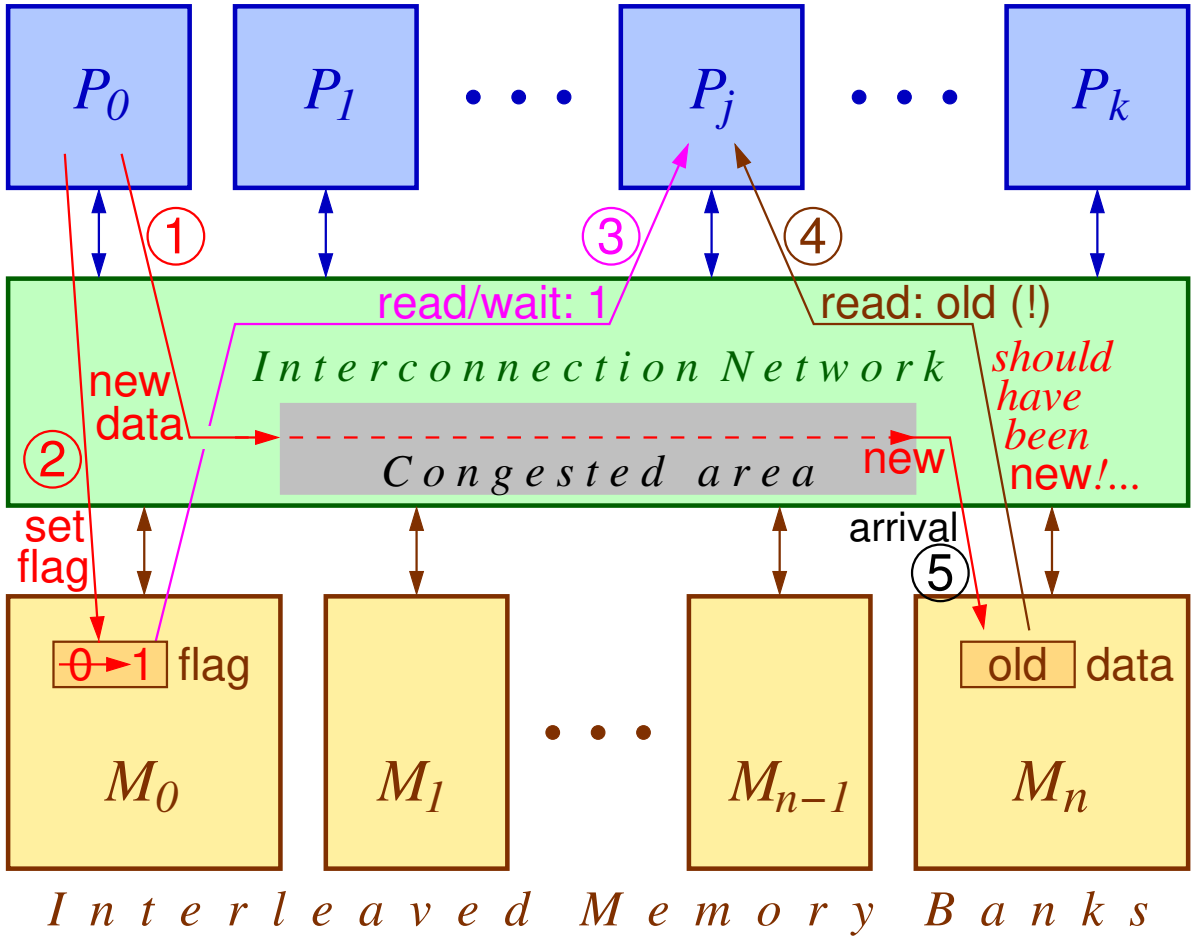
- Αναδιάταξη χρονικής αλληλουχίας λειτουργιών (π.χ. pipelining, scheduling) ελεύθερη για διαφορετικές διευθύνσεις. Ομοίως για ίδια διεύθυνση με σεβασμό των RAR, RAW, WAR, WAW

I/O / Communication Space



- Η χρονική αλληλουχία έχει σημασία, ακόμη και για προσπελάσεις σε διαφορετικές διευθύνσεις
  - Παρενέργειες (Side-Effects)
  - Συγχρονισμός / Σηματοδοσία (signaling) αλλαγών δεδομένων
  - “Memory Consistency” («Συνέπεια Μν.»)

Memory Consistency – Συνέπεια Μνήμης



- Όταν επιτρέπονται λειτουργίες εκτός σειράς (out-of-order), για αύξηση επίδοσης, ο απλοϊκός συγχρονισμός δεν επαρκεί
- Τάση προς *Release Consistency*, με προσθήκη εντολών *Memory Fence* (*Memory Barrier*)

Interrupt-driven I/O (Ε/Ε βάσει διακοπών)

- Το Polling συχνά είναι ασύμφορα χρονοβόρο
  - ιδιαίτερα όταν ρυθμός άφιξης ποικίλλει ευρέως (απρόβλεπτοι χρόνοι)
  - ανάγνωση I/O Status reg. 10-100'δες κύκλους ρολογ. (non-cacheable!)
- I/O Interrupts: ο συνήθης εναλλακτικός μηχανισμός
  - Interrupt (Διακοπή) παρόμοια με Exception (Εξαίρεση) – §12.3
  - Ασυσχέτιστη με το πρόγραμμα που τρέχει όταν αυτή συμβαίνει
  - Δεν υπάρχει ανάγκη ακύρωσης καμιάς εντολής στην pipeline
  - Απλός μηχανισμός: ένα σύρμα ανά περιφερειακή συσκευή, το OR όλων τους προκαλεί την αποθήκευση & αλλαγή του PC κλπ.
  - Συχνά με προτεραιότητες ⇒ ενδεχόμενη συσσώρευση διακοπών χαμηλότερης προτεραιότητας ⇒ προσοχή στον καταχ. CAUSE

Χρονικό κόστος Διακοπών και Δειγματοληψίας

- Παρ’ ότι η εξαίρεση/διακοπή μοιάζει με άλμα, εντούτοις…

- Σχεδόν πάντα αυτή κοστίζει ~1000 κύκλους ρολογιού (!)
  - ένα μικρό μυστήριο το γιατί ακόμα ισχύει αυτό στη μεγάλη πλειοψηφία, παρ’ όλο που πολλοί προσπάθησαν να το μειώσουν
  - μάλλον το άθροισμα κάμποσων παραγόντων, κυρίως:
  - σώσιμο/επαναφορά καταχωρητών διακοπείσας διεργασίας
  - αναζήτηση αιτίας / “housekeeping”
  - αστοχίες κρυφών μνημών και TLB

- Δειγματοληψίες επίσης κοστίζουν: non-cacheable I/O reg.
  - μία τεχνική: όταν διακοπή από real-time clock (~50-120 Hz), τότε το Λειτουργικό δειγματοληπτεί πολλές περιφερειακές μαζί

Ταχείες συσκευές: απλός καταχωρητής δεν αρκεί

- Π.χ. μία γραμμή δικτύου μέτριας ταχύτητας ή ένας (μόνον) δίσκος:
  - 1 Gbit/s ≈ 100 MBy/s ⇒ 1 bit / ns ⇒ 32 bits = 1 data reg. κάθε 32 ns
  - Polling: read the Status Reg. (non-cacheable, off-chip) ≈ 1 DRAM access ≈ 100 ns (κι άλλα τόσα για Data Reg.) >> 32 ns που φτάνει κάθε λέξη!
  - Interrupts: απλή είσοδος & έξοδος kernel Interrupt handler ≈ 1000-2000 κύκλοι ρολογιού ≈ 1 μs >> 32 ns που φτάνει κάθε λέξη!

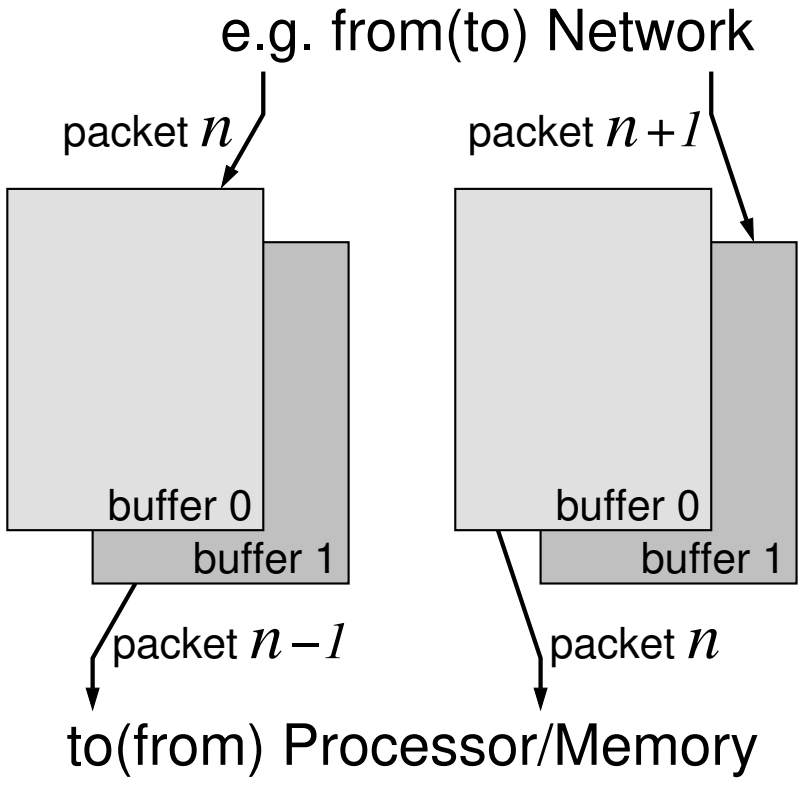
⇒ Το κόστος εκκίνησης του Interrupt πρέπει να αποσβεστεί μεταφέροντας πολλές λέξεις –όχι μόνον μία– μετά από κάθε Interrupt

⇒ Χρειαζόμαστε έναν ολόκληρο Data Buffer (π.χ. 4 KBytes), αντί έναν μόνο Data Register, στην Περιφερειακή συσκευή

- μόνον 1 Interrupt για κάθε 1024 λέξεις, δηλ. κάθε 32 μs
- και πάλι πρόβλημα εάν ≈100ns το κάθε load λέξης από τον Data Buffer

Double Buffering: Διαχείριση παράλληλης πρόσβασης

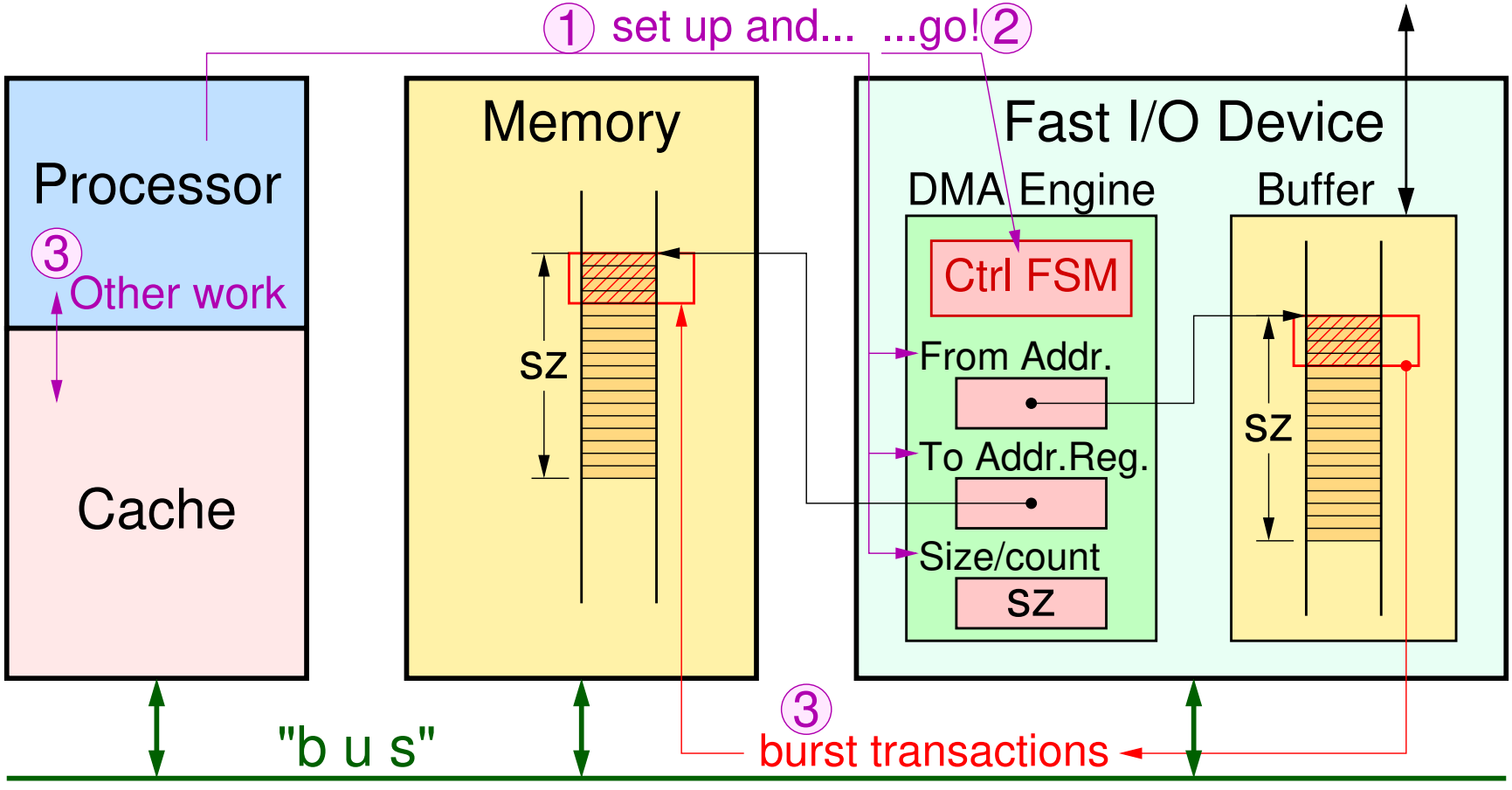
- *Δεν* απαιτείται δίπορτη μνήμη
- Δύο “banks” μονόπορτης μνήμης
  - Εάν δεν αρκεί η παροχή μίας “bank”
  - Όπως η Διαφύλλωση, αλλά με διαφορετική διασπορά διευθύνσεων
- Χρήσιμο και για την διαχείριση χώρου και χρόνου των 2 φάσεων:
  - άφιξη (ή αναχώρηση) από πλ. I/O
  - παραλαβή (ή προετοιμασία) από πλευράς επεξεργαστή
- Σαν το pipelining, αλλά χωρίς αντιγραφή/μετακίνηση data



Ποιος μεταφέρει Data μεταξύ Συσκευής και Μνήμης;

- Εντελώς ασύμφορο μέσω λογισμικού στον επεξεργαστή για μεγάλο όγκο και ρυθμό μεταφοράς δεδομένων
  - βρόχος με load από περιφεριακή, store στη μνήμη (ή αντίστροφα)
  - περιφερειακή: θέσεις non-cacheable ⇒ ~100+ κύκλοι κάθε load/store
  - μόνο μία λέξη ανά load/store: δεν αξιοποιεί την οικονομία κλίμακας της αρτηρίας επεξεργαστή-περιφερειακών όποτε μεταφέρονται πολλές συνεχόμενες λέξεις όλες μαζί
- Πολύ απλή διαδικασία η αντιγραφή data (κατά ομάδες)
  - Μπορεί να την κάνει και μία απλή FSM, σε hardware
  - αφήνοντας τον επεξεργαστή διαθέσιμο για πιο πολύπλοκες εργασίες
  - “Direct Memory Access” (DMA) – «Απευθείας Πρόσβαση Μνήμης»

Direct Memory Access (DMA)



DMA versus Caching

- <to be done>