Ενσωματωμένη Αρχιτεκτονική

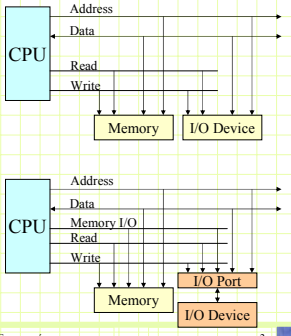
και

Προγραμματισμός Επεξεργαστών

I/O, Timers, Buffering και DMA

CPU - Bus - I/O

- Η CPU επικοινωνεί με I/O συσκευές όπως πληκτρολόγιο, ποντίκι, video, network, μονάδες δίσκου, LEDs
- Memory-mapped I/O
  - ✓ Οι συσκευές είναι τοποθετημένες σε συγκεκριμένες τοποθεσίες μνήμης (memory locations) όπως η RAM
  - ✓ Uses load/store instructions just like accesses to memory
- Ported I/O
  - ✓ Ειδικές bus γραμμές και οδηγίες

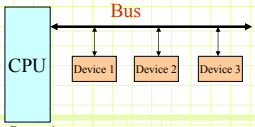


I/O Register Basics

- Οι I/O καταχωρητές **ΔΕΝ** είναι σαν τη κανονική μνήμη
  - ✓ Τα γεγονότα των συσκευών μπορούν να αλλάξουν τις τιμές τους (π.χ., status registers)
  - ✓ Διαβάζοντας ένα καταχωρητή μπορεί να αλλάξει την τιμή του (π.χ., error condition reset)
    - o Για παράδειγμα, μην περιμένεις να πάρεις την ίδια τιμή αν τον διάβασες δύο φορές
  - ✓ Μερικοί είναι διάβασε-μόνο (read-only) (π.χ., receive registers)
  - ✓ Μερικοί είναι γράψε-μόνο (write-only) (π.χ., transmit registers)
  - ✓ Μερικές φορές πολλαπλοί I/O καταχωρητές τοποθετούνται (mapped) στην ίδια διεύθυνση
    - o Επιλογή του ενός βασισμένη σε άλλη πληροφορία(π.χ., διάβασε vs. γράψε ή επιπλέον bits ελέγχου)
- Τα bits σε καταχωρητή ελέγχου συχνά το καθένα διευκρινίζει κάτι διαφορετικό και σημαντικό -- και έχει σημαντικές παρενέργειες
- Η Cache πρέπει να απενεργοποιείται για memory-mapped διευθύνσεις
- Όταν χρησιμοποιεί I/O καταχωρητές, πρέπει να πει στον compiler ότι η τιμή μπορεί να αλλάξει από μόνη της (`volatile int *ptr;`)

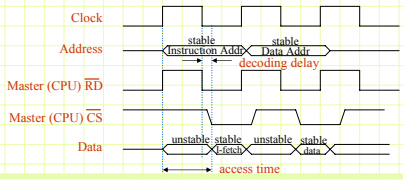
Bus Protocols

- Το προτόκολο αναφέρεται στο σύνολο κανόνων που έχει συμφωνηθεί και από τον bus master και από τον bus slave
  - ✓ Synchronous bus - οι μεταφορές συμβαίνουν σε σχέση με τις διαδοχικές ακμές ενός ρολογιού
  - ✓ Asynchronous bus - η μεταφορά δεν ακολουθεί κάποια συγκεκριμένη σχέση συγχρονισμού
  - ✓ Semi-synchronous bus - Οι διαδικασίες/έλεγχος γίνονται ασύγχρονα, αλλά η μεταφορά δεδομένων γίνεται συγχρόνως



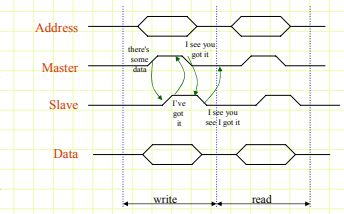
Synchronous Bus Protocol

- Η μεταφορά γίνεται σε σχέση με τις διαδοχικές ακμές του ρολογιού του συστήματος
- Παράδειγμα:
  - ✓ Η διεύθυνση μνήμης τοποθετείται στο bus διευθύνσεων σε ορισμένο χρόνο, σχετικό με την αυξανόμενη ακμή του ρολογιού
  - ✓ Από τη σταθερή ακμή αυτού του παλμού του ρολογιού, οι πληροφορίες διευθύνσεων έχουν το χρόνο να σταθεροποιηθούν, έτσι η ΔΙΑΒΑΣΜΕΝΗ γραμμή βεβαιώνεται
  - ✓ Μόλις το chip επιλεγεί, τότε η μνήμη μπορεί να τοποθετήσει τα περιεχόμενα της συγκεκριμένης τοποθεσίας στο bus δεδομένων



Asynchronous Bus Protocol

- ***ΔΕΝ*** χρησιμοποιείται ρολόι του συστήματος
- Χρήσιμο για συστήματα που η CPU και οι I/O συσκευές τρέχουν με διαφορετικές ταχύτητες
- *Παράδειγμα*
  - Ο Master βάζει διεύθυνση και δεδομένα στο bus και τότε υψώνει το Master σήμα
  - Ο Slave βλέπει το master σήμα, διαβάζει τα δεδομένα και υψώνει το Slave σήμα
  - Ο Master βλέπει το Slave σήμα και κατεβάζει το Master σήμα
  - Ο Slave βλέπει το Master σήμα κατεβασμένο και κατεβάζει το Slave σήμα



Αποκαλούμε αυτή την ανταλλαγή "handshaking"

Bus Arbitration (Διαιτησία)

- Τι συμβαίνει όταν πολλές συσκευές θέλουν να έχουν πρόσβαση στο bus?
- **Σχέδιο 1**: Κάθε συσκευή συνδέεται στη bus request line (λίστα αιτήματος) και το πρώτο εκεί το χρησιμοποιεί
- **Σχέδιο 2**: daisy chain the devices - devices further down the daisy chain pass the request to the CPU - device's priority decreases further down the daisy chain
- **Σχέδιο 3**: μία λίστα αιτήματος bus για κάθε bus και ο διαιτητής εφαρμόζει την πολιτική διαιτησίας για να αποφασίσει ποιος θα χρησιμοποιήσει το bus μετά

Serial Communications Protocols

- Το **Communications protocol (προτόκολο επικοινωνίας)** είναι σύμβαση για τη μετάδοση στοιχείων που περιλαμβάνει λειτουργίες όπως ο συγχρονισμός, η μορφοποίηση και η αντιπροσώπευση στοιχείων
- Δύο κατηγορίες προτοκόλων:

Asynchronous protocols (ασύγχρονα προτόκολα)

✓ Διαδοχικά δεδομένα εμφανίζονται στη ροή δεδομένων σε αυθαίρετους χρόνους, χωρίς συγκεκριμένο έλεγχο ρολογιών που διαχειρίζεται τις σχετικές καθυστερήσεις στα δεδομένα

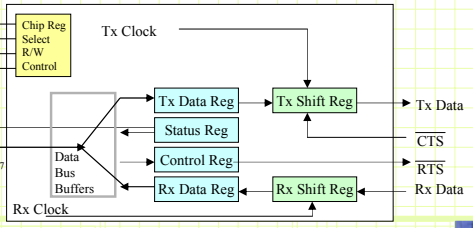
Synchronous protocols (σύγχρονα προτόκολα)

✓ Κάθε διαδοχικό δεδομένο στη ροή των δεδομένων διαχειρίζεται από ένα κύριο ρολόι δεδομένων (master data clock) και εμφανίζεται σε ένα συγκεκριμένο διάστημα του χρόνου

- Συχνά, τα πρωτόκολλα παραδίδουν συριακά δεδομένα με 8-bit χαρακτήρες -- τα ασύγχρονα πρωτόκολλα συμπεριφέρονται σε κάθε χαρακτήρα ως μεμονωμένο μήνυμα, και οι χαρακτήρες εμφανίζονται στη ροή δεδομένων στε αυθαίρετους σχετικά χρόνους. Εντούτοις, μέσα σε κάθε χαρακτήρα, τα bits εκπέμπονται με σταθερό προκαθορισμένο ρυθμό ρολογιού

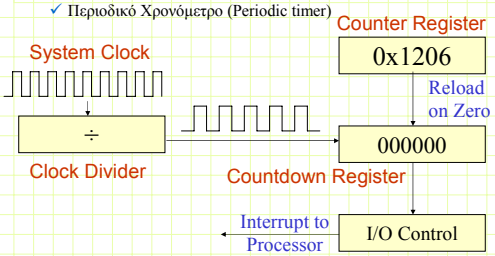
Διασυνδέοντας σειριακά δεδομένα στο μικροεπεξεργαστή

- Ο επεξεργαστής έχει παράλληλα buses για τα δεδομένα -- χρειάζεται να μετατραπούν τα σειριακά δεδομένα σε παράλληλα (και αντίστροφα)
- Ο τυποποιημένος τρόπος είναι με UART
- UART - Universal asynchronous receiver and transmitter
  - ✓ USART - Universal synchronous and asynchronous receiver and transmitter



Timer

- Μια συσκευή που χρησιμοποιεί την είσοδο ρολογιών μεγάλων ταχυτήτων για να παρέχει σειρά χρόνου ή γεγονότων που βασίζονται στην αρίθμηση
  - ✓ Χρονόμετρο μιας ακμής (One shot timer)
  - ✓ Περιοδικό Χρονόμετρο (Periodic timer)



Interrupt vs. Polled I/O

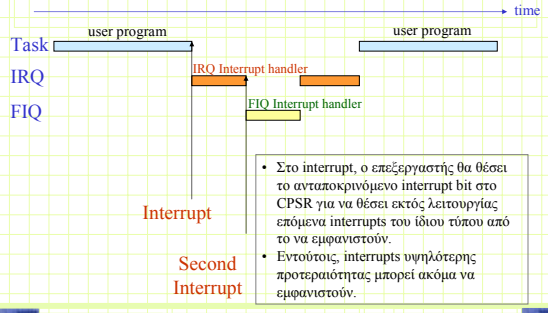
- Το Polled I/O απαιτεί ηCPU να *ρωτήσει* μια συσκευή (π.χ. *toggle switches*) εάν η συσκευή απαιτεί εξυπηρέτηση
  - ✓ Για παράδειγμα, εάν οι διακόπτες αναστροφής (toggle switches) έχουν αλλάξει τη θέση
  - ✓ Το λογισμικό σχεδιάζει για την ψήφιση των συσκευών (polling the devices) και γράφεται για να ξέρει πότε μια συσκευή θα εξυπηρετηθεί
- Το Interrupt I/O επιτρέπει τη συσκευή να *διακόψει (interrupt)* τον επεξεργαστή, αναγγέλλοντας ότι η συσκευή απαιτεί την προσοχή
  - ✓ Αυτό επιτρέπει στην CPU να αγνοήσει τις συσκευές εκτός αν ζητούν συντήρηση (μέσω interrupts)
  - ✓ Το λογισμικό δεν μπορεί να γνωρίζει πότε θα γίνει interrupts επειδή interrupts μπορεί να συμβεί οποιαδήποτε στιγμή -- επομένως, το λογισμικό δεν έχει καμία ιδέα πότε θα συμβεί interrupt
  - ✓ Αυτό το καθιστά δυσκολότερο να γραφεί τον κώδικας
- Οι επεξεργαστές μπορούν να προγραμματιστούν για να αγνοήσουν τα interrupts
  - ✓ Αυτό το αποκαλούμε ***κάλυψη (masking)*** των interrupts
  - ✓ Οι διαφορετικοί τύποι διακόπτουν μπορούν να καλυφθούν (IRQ vs. FIQ)

IRQ vs. FIQ

- Εξωτερικά interrupts που είναι ενεργά ΧΑΜΗΛΑ (LOW)
- FIQs έχει πιο μεγάλη προτεραιότητα από IRQs
  - ✓ Όταν πολλαπλά interrupts συμβαίνουν, τα FIQs εξυπηρετούνται πριν από τα IRQs
  - ✓ Εξυπηρετώντας ένα FIQ αναγκάζει το IRQs να τεθεί εκτός λειτουργίας έως ότου ο χειριστής FIQ (FIQ handler) να τους επανενεργοποιήσει
    - o CPSR απεκατεστημένος από το SPSR στο τέλος του FIQ handler
- Πως τα FIQs γίνονται γρηγορότερα?
  - ✓ Έχουν πέντε πρόσθετους καταχωρητές στη διάθεσή τους, επιτρέποντας τα να καταχωρούν allowing them to store δεδομένα ανάμεσα στις κλήσεις τους χειριστή (handler)
  - ✓ Το FIQ διάνυσμα είναι η τελευταία είσοδος στο διανυσματικό πίνακα
    - o Ο FIQ handler μπορεί να τοποθετηθεί άμεσα στη θέση του διανύσματος και να τρέξει διαδοχικά μετά από τη θέση
    - o Cache-based systems: Vector table + FIQ handler όλα κλειδωμένα σε ένα block

Nested/Re-entrant Interrupts

- Interrupts can occur within interrupt handlers



- Στο interrupt, ο επεξεργαστής θα θέσει το ανταποκρινόμενο interrupt bit στο CPSR για να θέσει εκτός λειτουργίας επόμενα interrupts του ίδιου τύπου από το να εμφανιστούν.
- Εντούτοις, interrupts υψηλότερης προτεραιότητας μπορεί ακόμα να εμφανιστούν.

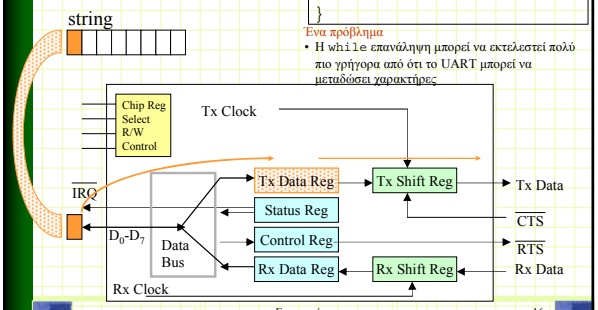
Drivers (Serial I/O example)

- High-level I/O call
  - ✓ `printf("the number is %d\n", someNumber);`
- Low-level details
  - ✓ `printf()` είναι μια κλήση βιβλιοθηκών που μορφοποιεί την έξοδο (e.g., converts `%d` formats) και έπειτα κάνει το σύστημα να καλέσει στην έξοδο τη μορφοποιημένη συμβολοσειρά
    - o Η μορφοποιημένη συμβολοσειρά δεν είναι τίποτα περισσότερο από μια σειρά χαρακτήρων
    - o Οι Low-level ρουτίνες τότε παράγουν τη συμβολοσειρά, ένα χαρακτήρα τη φορά χρησιμοποιώντας UART

Drivers (Serial I/O example) (cont'd)

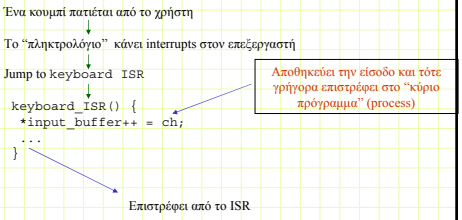
```
while (*string != '\0'){
    if (UART is empty()
        printChar(*string++);
}
```

Ένα πρόβλημα
- Η `while` επανάληψη μπορεί να εκτελεστεί πολύ πιο γρήγορα από ότι το UART μπορεί να μεταδώσει χαρακτήρες



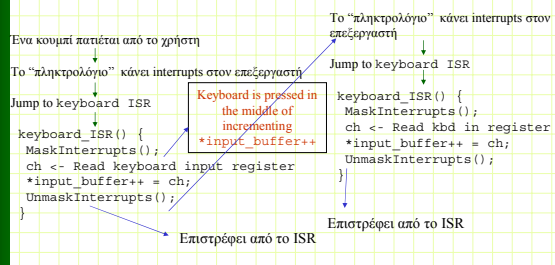
Drivers (keyboard example)

- Πρόσθεσε ένα buffer (στο λογισμικό ή το υλικό) για τους χαρακτήρες εισαγωγής.
  - Αυτό αποσυνδέει το χρόνο για την επεξεργασία από το χρόνο μεταξύ των πληκτρολογήσεων, και παρέχει μία ανώτερη υπολογιστική δέσμευση στο χρόνο που απαιτείται για να εξυπηρετήσει ένα interrupt πληκτρολογίου .



Drivers (keyboard example) (cont'd)

- Εάν τα interrupts είναι καλυμμένα (IRQ και FIQ απενεργοποιούνται), τίποτα δεν θα επεξεργαστεί έως ότου ολοκληρώνει το ISR και επιστρέφει.
  - Θυμήσου: η είσοδος των IRQs τρόπων καλύπτει τα IRQ και η είσοδος του FIQ καλύπτει τα IRQs και τα FIQs



Critical Sections of Code

- Κομμάτια κώδικα που πρέπει να εμφανιστούν ως **ατομική δράση**

```
printStr(*string)          printStr("this is a line");
char *string;
{
   MaskInterrupts();
   while (*string){
      outputBuffer[tail++] = *string++;
   }
   UnmaskInterrupts();
}
```

T H I S I S

tail points here - and a timer interrupt occurs

Άλμα σε `timer_ISR` συμβαίνει αφού το `printStr()` ολοκληρωθεί

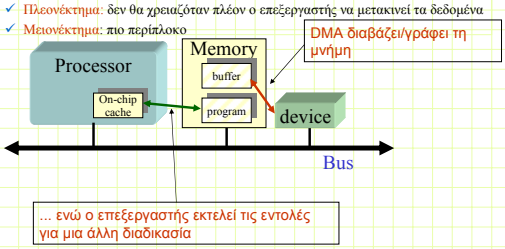
```
timer_ISR(){
   clockTicks++;
   printStr(convert(clockTicks));
}
```

T H I S I S A L I N E

Ατομική δράση - δράση που "φαίνεται" οτι πραγματοποιείται σε μια **ενιαία λειτουργία**

Direct Memory Access (DMA)
Άμεση πρόσβαση μνήμης

- Αντί για προγραμματισμένο I/O μπορούμε να κάνουμε την συσκευή λίγο εξυπνότερη
  - ✓ Να κάνουμε την συσκευή ικανή να μετακινεί δεδομένα στην/από τη μνήμη την ίδια
  - ✓ Πλεονέκτημα: δεν θα χρειαζόταν πλέον ο επεξεργαστής να μετακινεί τα δεδομένα
  - ✓ Μειονέκτημα: πιο περίπλοκο



... ενώ ο επεξεργαστής εκτελεί τις εντολές για μια άλλη διαδικασία

DMA

- Η συσκευή μπορεί "να κλέψει" κύκλους πρόσβασης της μνήμης από το bus ενώ ο επεξεργαστής δεν διαβάζει ή δεν γράφει
- Γενικά μία "block" κίνηση ρυθμίζεται -- ίσος στα 512 bytes ανά block
  - ✓ Η ρυθμισμένη ρουτίνα χρησιμοποιείται για να ξεκινήσει η μεταφορά
  - ✓ Η DMA μεταφορά έχει ολοκληρωθεί
  - ✓ Ένα bit σήματος στέλνεται να επιβεβαιώσει την ολοκλήρωση της μεταφοράς.
- Υπάρχουν άλλες DMA μέθοδοι
  - ✓ Μερικές φορές υπάρχει μια DMA-only συσκευή που μεταφέρει δεδομένα σε/από άλλες συσκευές