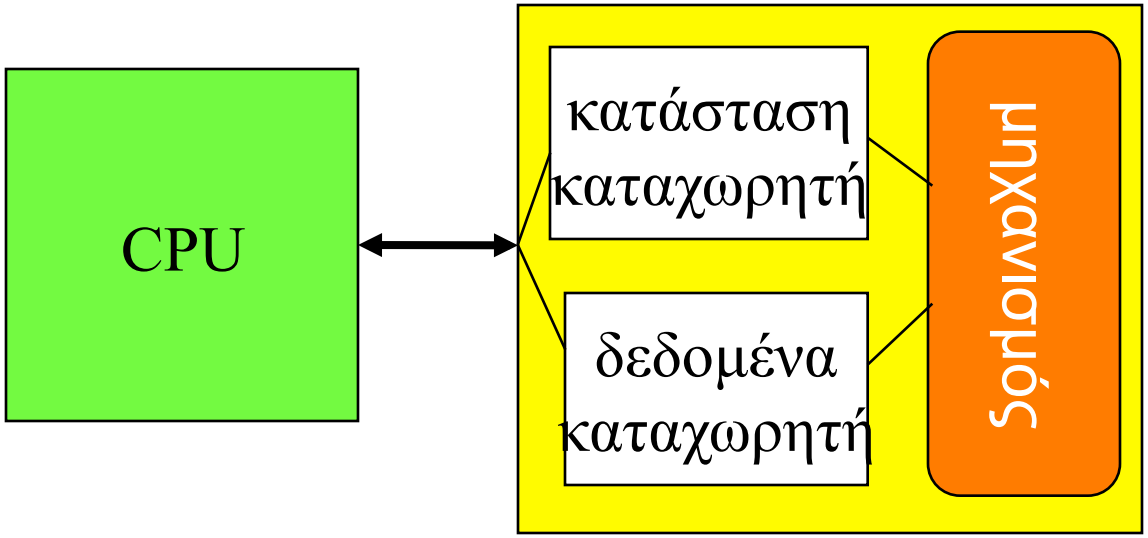
CPUs (Κεντρικές Μονάδες Επεξεργασίας)

- Είσοδος και έξοδος.
- Λειτουργία υπερχρήστη, εξαιρέσεις, παγίδες.
- Συνεπεξεργαστές.

Συσκευές εισόδου/εξόδου (I/O)

- Συνήθως περιλαμβάνουν κάποια μη ψηφιακά εξαρτήματα.
- Τυπική ψηφιακή πρόσβαση στη CPU:

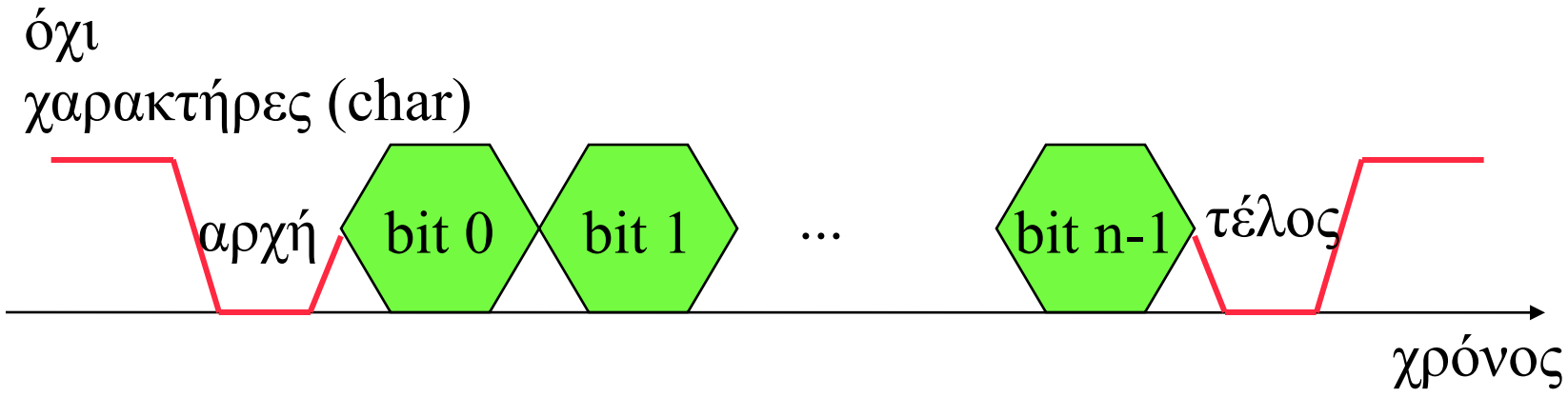


Εφαρμογή: 8251 UART

- Ασύγχρονος πομποδέκτης γενικής χρήσης - Universal asynchronous receiver transmitter (UART) : παρέχει σειριακή επικοινωνία.
- Οι συναρτήσεις του 8251 ενσωματώνονται σε ένα τυπικό ολοκληρωμένο διασύνδεσης με PC.
- Επιτρέπει σε πολλές παραμέτρους επικοινωνίας να προγραμματιστούν.

Σειριακή επικοινωνία

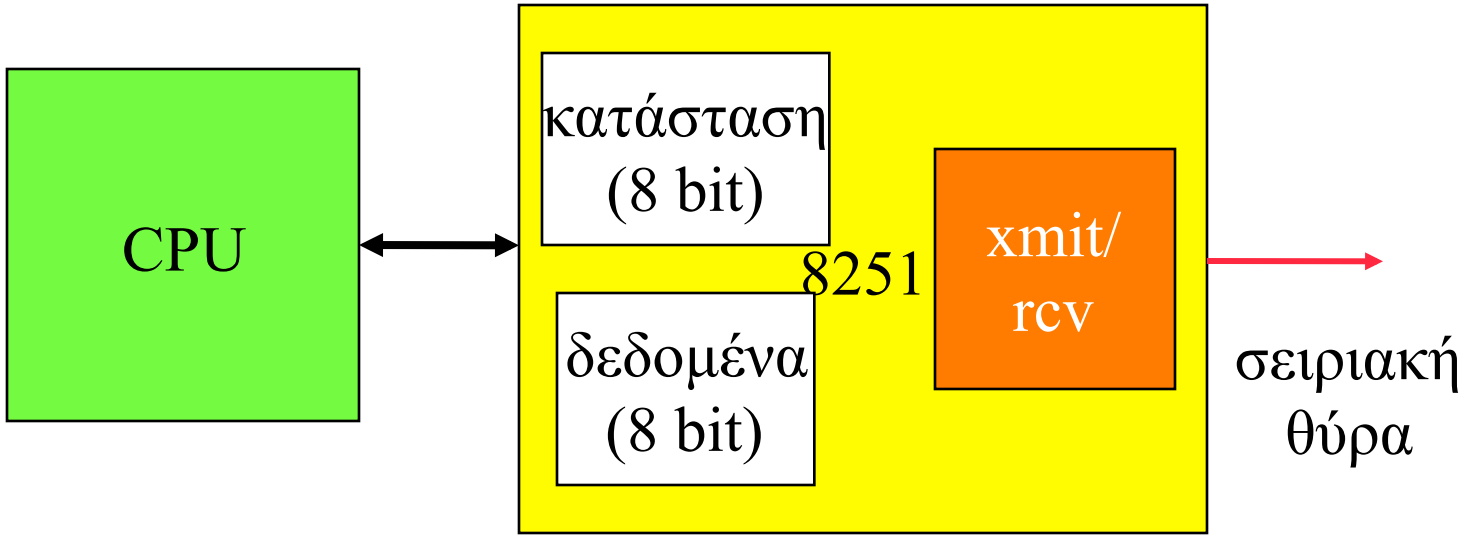
- Οι χαρακτήρες μεταδίδονται χωριστά:



Παράμετροι σειριακής επικοινωνίας

- Ρυθμός μετάδοσης δεδομένων - Baud (bit) rate.
- Αριθμός bits ανά χαρακτήρα.
- Ισοτιμία/όχι ισοτιμία.
- Άρτια/περιττή ισοτιμία.
- Μήκος του bit παύσης (1, 1.5, 2 bits).

8251: διασύνδεση με CPU



Προγραμματισμός εισόδων/ εξόδων (I/O)

- Δύο τύποι εντολών μπορούν να υποστηρίξουν εισόδους/ εξόδους :
  - Εντολές εισόδου/εξόδου ειδικού σκοπού.
  - Εντολές φόρτωσης/αποθήκευσης απεικονισμένες στη μνήμη.
- Οι επεξεργαστές Intel x86 παρέχουν `in`, `out` εντολές. Οι περισσότερες άλλες CPUs χρησιμοποιούν εισόδους/εξόδους απεικονισμένες στη μνήμη.
- Οι εντολές εισόδου/εξόδου ειδικού σκοπού δεν αποκλείουν εισόδους/εξόδους απεικονισμένες στη μνήμη.

Οι είσοδοι/έξοδοι απεικονισμένες στη μνήμη στον ARM

- Ορισμός της θέσης της συσκευής:

```
DEV1 EQU 0x1000
```

- Κώδικας ανάγνωσης/εγγραφής:

```
LDR r1,#DEV1 ; καθορίζει τις διευθύνσεις
               της συσκευής
LDR r0,[r1] ; διαβάζει το DEV1
LDR r0,#8 ; καθορίζει την τιμή που θα
            γράψει
STR r0,[r1] ; γράφει την τιμή στη συσκευή
```

Οι είσοδοι/έξοδοι απεικονισμένες στη μνήμη στον SHARC

- Η συσκευή πρέπει να είναι στον εξωτερικό χώρο μνήμης. (πάνω από το 0x400000).
- Γίνεται χρήση DM για τον έλεγχο πρόσβασης:

```
I0 = 0x400000;
M0 = 0;
R1 = DM(I0,M0);
```

Peek and poke

- Παραδοσιακή διασύνδεση σε γλώσσες υψηλού επιπέδου:

```
int peek(char *location) {
  return *location; }

void poke(char *location, char
  newval) {
  (*location) = newval; }
```

Έξοδος αναμονής λόγω απασχόλησης (busy/wait)

- Ο απλούστερος τρόπος προγραμματισμού μιας συσκευής.
  - Χρήση εντολών για να εξετάζεται η ετοιμότητα της συσκευής.

```
current_char = mystring;
while (*current_char != '\0') {
  poke(OUT_CHAR,*current_char);
  while (peek(OUT_STATUS) != 0);
  current_char++;
}
```

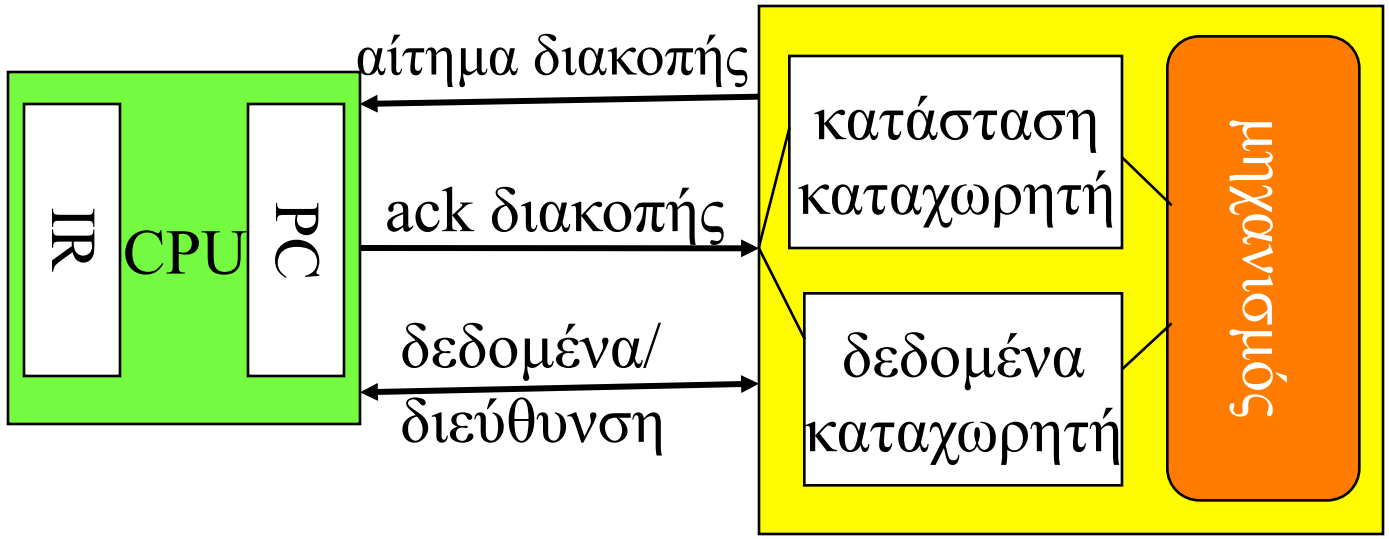
Ταυτόχρονη είσοδος/έξοδος αναμονής λόγω απασχόλησης

```
while (TRUE) {
  /* read */
  while (peek(IN_STATUS) == 0);
  achar = (char)peek(IN_DATA);
  /* write */
  poke(OUT_DATA,achar);
  poke(OUT_STATUS,1);
  while (peek(OUT_STATUS) != 0);
  }
```

Είσοδοι/έξοδοι με διακοπές (Interrupt I/O)

- Η αναμονή λόγω απασχόλησης(busy/wait) είναι πολύ αναποτελεσματική.
  - Η CPU δε μπορεί να εκτελέσει άλλη εργασία όταν εξετάζει τη συσκευή.
  - Δύσκολα γίνεται ταυτόχρονη είσοδος/έξοδος.
- Οι διακοπές επιτρέπουν σε μια συσκευή να αλλάξει τη ροή ελέγχου σε μια CPU.
  - Προκαλεί την κλήση υπορουτίνας για τη διαχείριση της συσκευής.

Διασύνδεση με διακοπές



Συμπεριφορά του interrupt

- Βασίζεται σε μηχανισμό κλήσης υπορουτινών.
- Η διακοπή υποχρεώνει την επόμενη εντολή να είναι κλήση υπορουτίνας σε προκαθορισμένη θέση.
  - Η διεύθυνση επιστροφής (PC) σώζεται για να συνεχιστεί από το σημείο διακοπής η εκτέλεση του προγράμματος που υπήρχε στο προσκήνιο.

Φυσική διασύνδεση της διακοπής

- Η CPU και η συσκευή συνδέονται με το δίαυλο της CPU.
- Χειραψία (handshake) CPU και συσκευής:
  - Η συσκευή επιβεβαιώνει την αίτηση για διακοπή.
  - Η CPU επιβεβαιώνει λήψη της διακοπής όταν μπορεί να το διαχειριστεί.

Παράδειγμα: χειριστής χαρακτήρα εισόδου/εξόδου

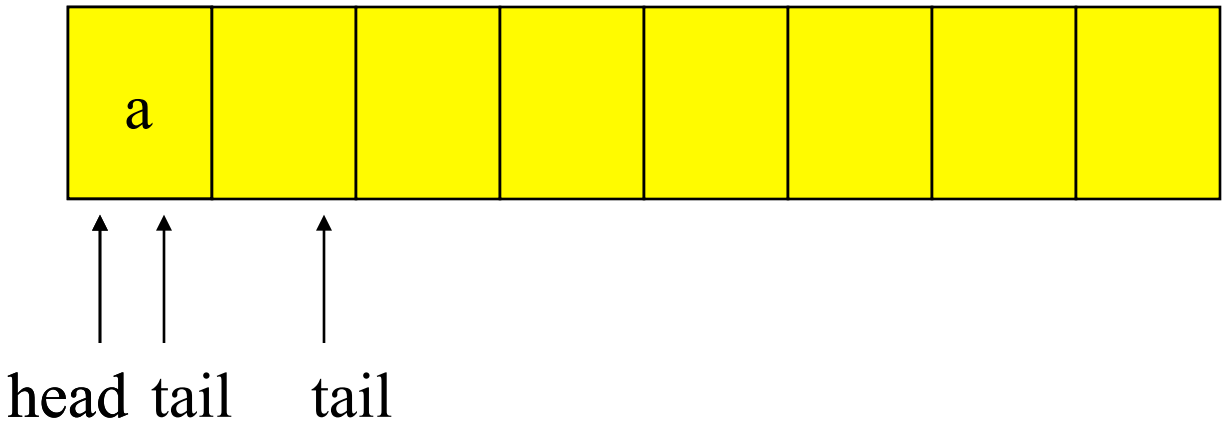
```
void input_handler() {
  achar = peek(IN_DATA);
  gotchar = TRUE;
  poke(IN_STATUS,0);
}
void output_handler() {
}
```

Παράδειγμα : κυρίως πρόγραμμα οδηγούμενο από διακοπές

```
main() {
  while (TRUE) {
     if (gotchar) {
          poke(OUT_DATA,achar);
          poke(OUT_STATUS,1);
          gotchar = FALSE;
          }
     }
  }
```

Παράδειγμα : διακοπή εισόδου/ εξόδου με buffers

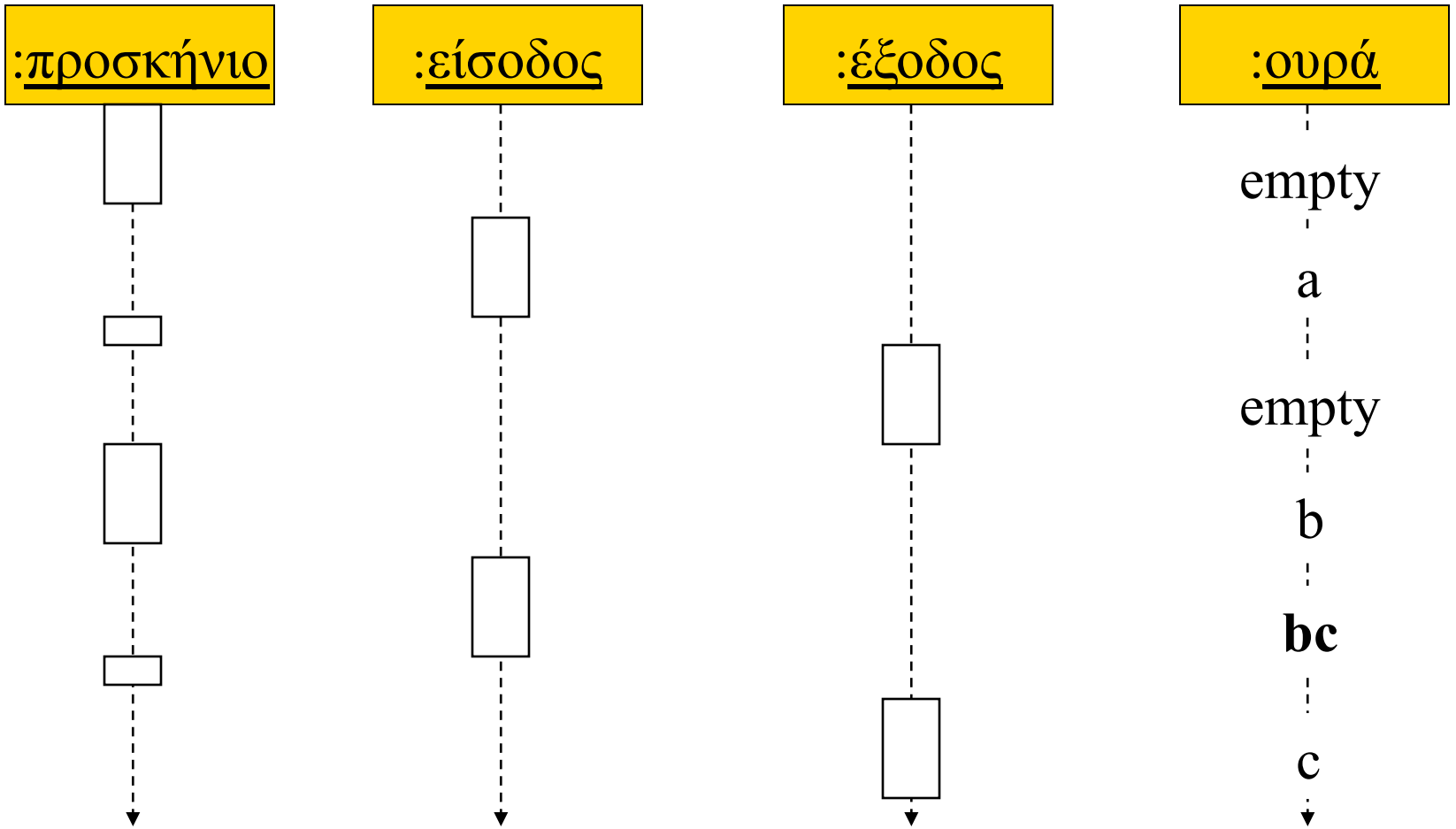
- Ουρές για χαρακτήρες:



Χειριστής εισόδου βασισμένος σε buffer

```
void input_handler() {
  char achar;
  if (full_buffer()) error = 1;
  else { achar = peek(IN_DATA);
  add_char(achar); }
  poke(IN_STATUS,0);
  if (nchars == 1)
     { poke(OUT_DATA,remove_char());
     poke(OUT_STATUS,1); }
  }
```

Διάγραμμα ακολουθίας εισόδου/εξόδου



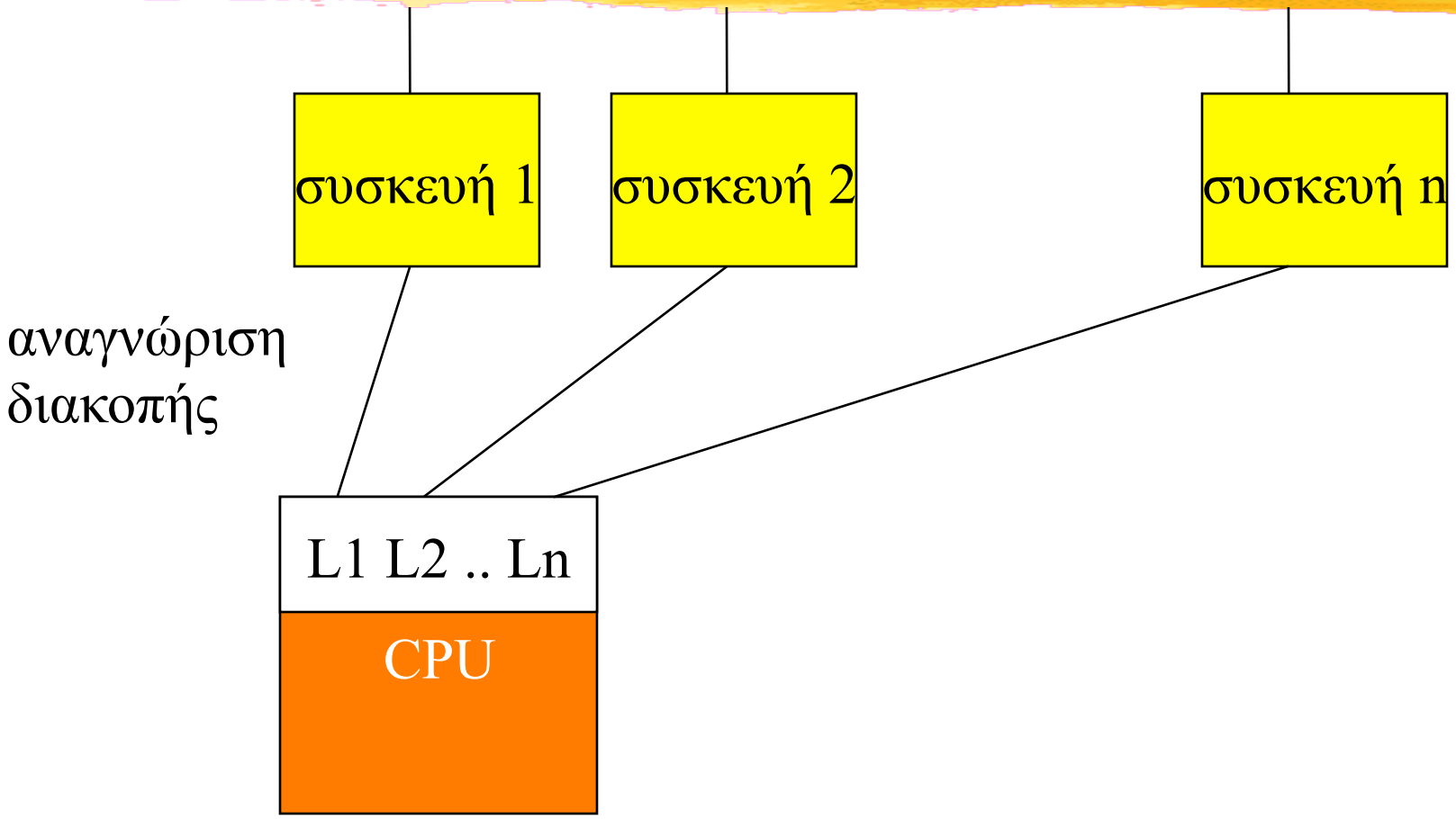
Διορθώνοντας σφάλματα στον κώδικα για διακοπές (debugging)

- Κι αν ξεχάσουμε να αλλάξουμε καταχωρητές
  - Το πρόγραμμα στο προσκήνιο μπορεί να παρουσιάζει μυστήρια σφάλματα.
  - Τα σφάλματα θα είναι δύσκολο να επαναληφθούν – θα εξαρτώνται από το συγχρονισμό της διακοπής.

Προτεραιότητες και διανύσματα

- Δύο μηχανισμοί μας επιτρέπουν να κάνουμε τις διακοπές πιο συγκεκριμένες:
  - Οι προτεραιότητες ορίζουν ποιος παίρνει τον έλεγχο της CPU πρώτο.
  - Τα διανύσματα ορίζουν τον κώδικα που καλείται για κάθε τύπο διακοπής.
- Οι μηχανισμοί είναι ορθογώνιοι: Οι περισσότερες CPUs παρέχουν και τα δύο.

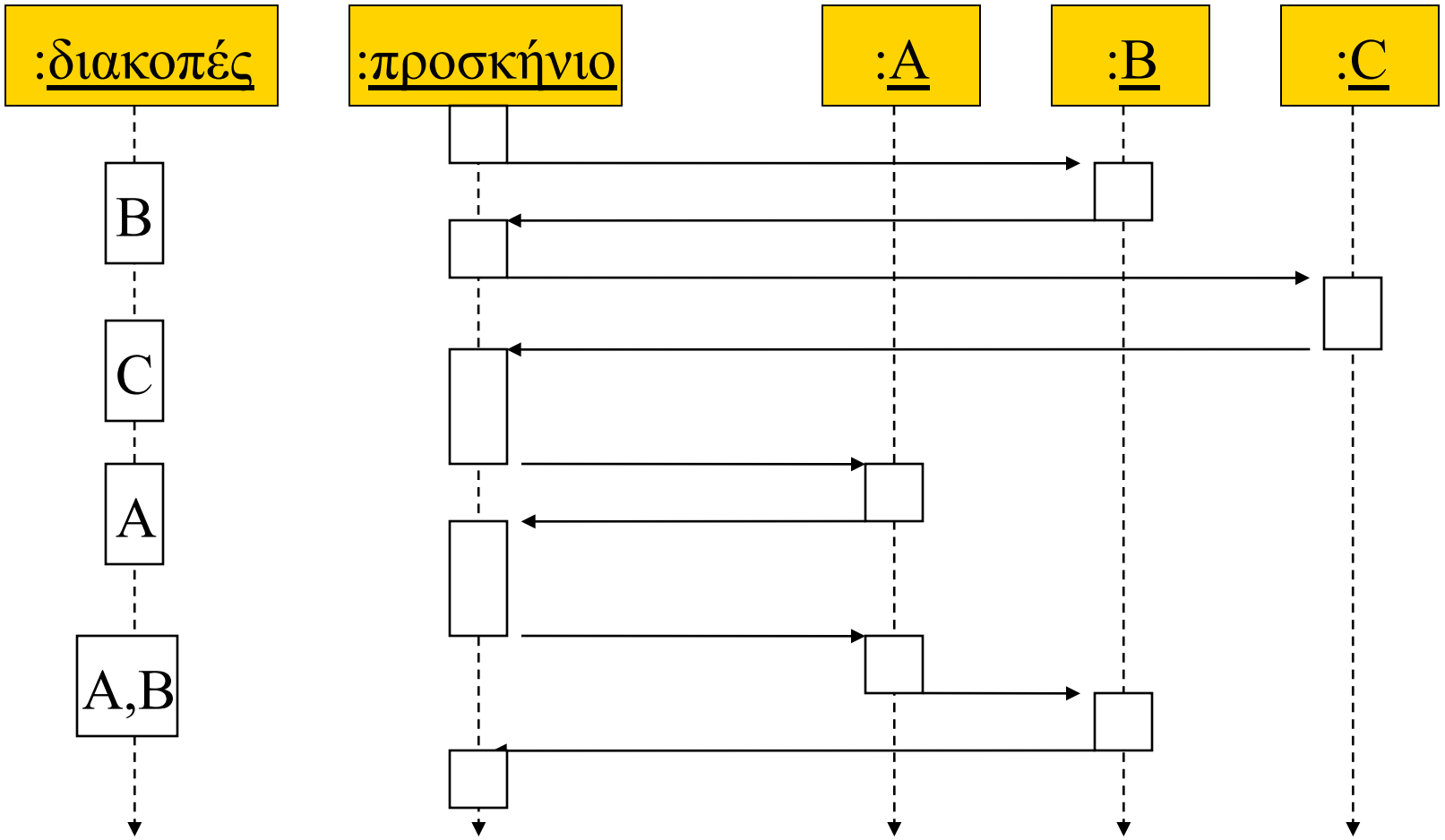
Διακοπές και προτεραιότητες



Διακοπή(Interrupt): ορισμός προτεραιοτήτων

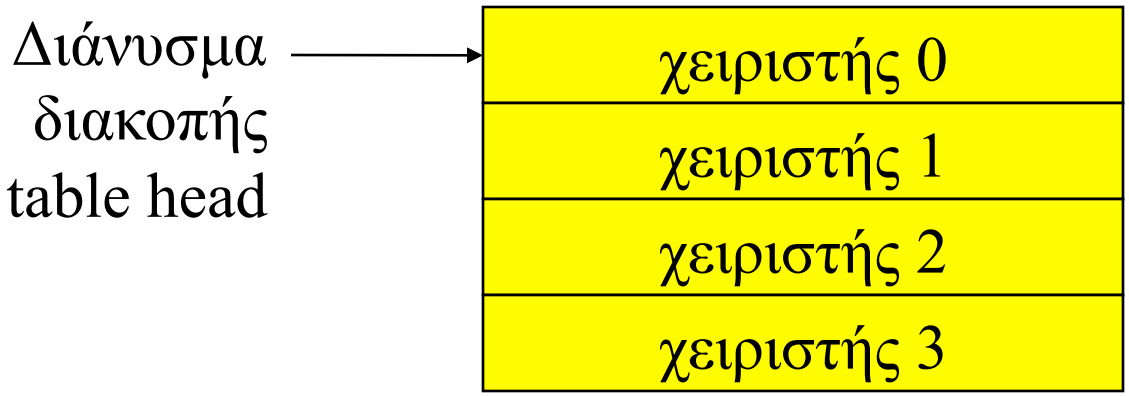
- Εφαρμογή μάσκας: Μία διακοπή με προτεραιότητα χαμηλότερη της τρέχουσας δεν αναγνωρίζεται μέχρι η τρέχουσα διακοπή να ολοκληρωθεί.
- Διακοπή χωρίς μάσκα (NMI): Στην υψηλότερη προτεραιότητα δεν εφαρμόζεται ποτέ μάσκα.
  - Συχνά χρησιμοποιείται για power down.

Παράδειγμα: I/O με προτεραιότητα

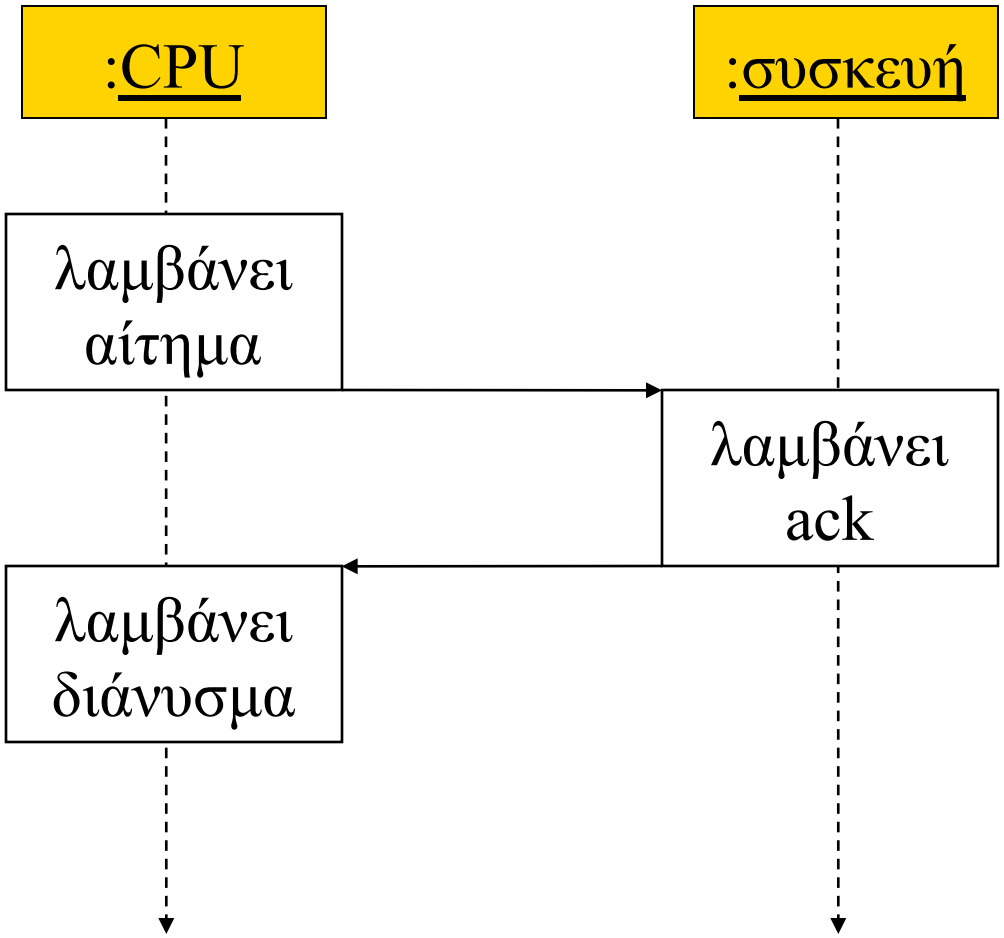


Διανύσματα διακοπής

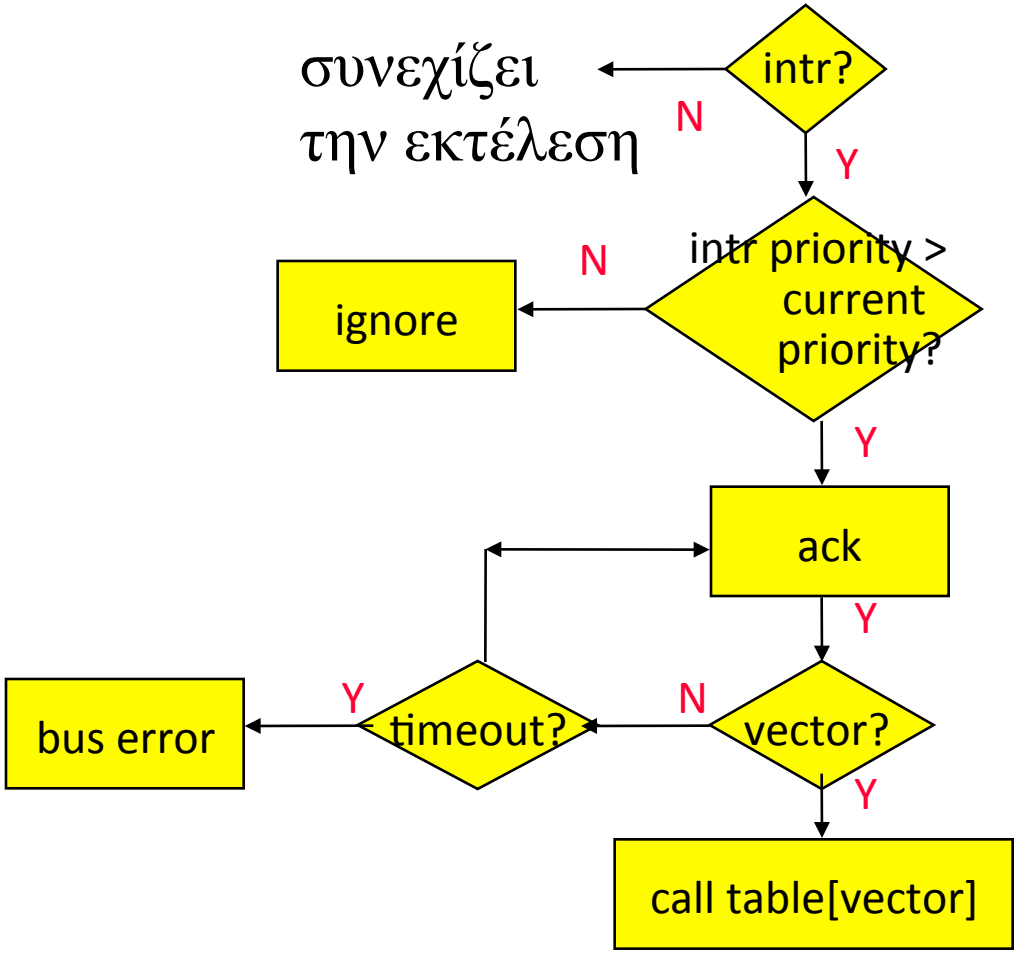
- Επιτρέπουν τη διαχείριση διαφορετικών συσκευών από διαφορετικό κώδικα.
- Πίνακας διανυσμάτων διακοπής:



Απόκτηση διανύσματος διακοπής



Γενικός μηχανισμός διακοπής



Ακολουθία διακοπής

- Η CPU επιβεβαιώνει την αίτηση.
- Η συσκευή στέλνει το διάνυσμα.
- Η CPU καλεί το διαχειριστή.
- Το λογισμικό διεκπαιραιώνει την αίτηση.
- Η CPU επαναφέρει την κατάσταση στο πρόγραμμα προσκηνίου.

Πηγές επιβάρυνσης από διακοπές

- Χρόνος εκτέλεσης της υπορουτίνας της διακοπής.
- Επιβάρυνση μηχανισμού διακοπής.
- Αποθήκευση/επανάκτηση καταχωρητών.
- Ποινές σχετικές με τη διοχέτευση.
- Ποινές σχετικές με την κρυφή μνήμη.

Διακοπές στον ARM

- Ο ARM7 υποστηρίζει δύο τύπους διακοπών:
  - Γρήγορες απαιτήσεις διακοπών (FIQs).
  - Απαιτήσεις διακοπών (IRQs).
- Ο πίνακας των διακοπών ξεκινά στη θέση 0.

Η διαδικασία των διακοπών στον ARM

- Ενέργειες της CPU:
  - Σώζει τον PC. Αντιγράφει τον CPSR στον SPSR.
  - Εξαναγκάζει τα bits στον CPSR να καταγράψουν τη διακοπή.
  - Εξαναγκάζει τον PC στο κατάλληλο διάνυσμα διακοπής (interrupt vector).
- Καθήκοντα του διαχειριστή:
  - Αποκαθιστά τον κατάλληλο PC.
  - Αποκαθιστά τον CPSR από τον SPSR.
  - Καθαρίζει τις σημαίες που εξουδετερώνουν τις διακοπές (interrupt disable flags).

Καθυστέρηση της διακοπής στον ARM

- Η χειρότερη περίπτωση καθυστέρησης (latency) για ανταπόκριση σε κάποιο interrupt είναι 27 κύκλοι:
  - Δύο κύκλοι για συγχρονισμό με το εξωτερικό αίτημα.
  - Έως 20 κύκλους για την ολοκλήρωση της τρέχουσας εντολής.
  - Τρεις κύκλοι για αποστολή των δεδομένων.
  - Δύο κύκλοι για είσοδο σε κατάσταση διαχείρισης του interrupt.

Δομή διακοπής στον SHARC

- Οι διακοπές διανυσματοποιούνται και ορίζονται σαν προτεραιότητα.
- Οι προτεραιότητες είναι σταθερές: υψηλότερη στο reset, χαμηλότερη στην SW διακοπή 3 από το χρήστη.
- Τα διανύσματα είναι επίσης σταθερά. Το διάνυσμα αποτυπώνεται στο πίνακα διανύσματος. Ο πίνακας ξεκινά στο 0x20000 στην εσωτερική μνήμη, και στο 0x40000 στην εξωτερική.

Ακολουθία διακοπής στον SHARC

Εκκίνηση: πρέπει να εκτελεί ή να είναι σε IDLE/IDLE16.

1. Εξαγωγή κατάλληλης διεύθυνσης στο διάνυσμα διακοπής.
2. Σπρώξιμο (push) της τιμής του PC στην PC στοίβα.
3. Γίνεται set το bit στον interrupt latch καταχωρητή.
4. Γίνεται set ο IMASKP στην παρούσα εμφωλευμένη κατάσταση.

Επιστροφή από διακοπή στον SHARC

Ξεκινά με την εντολή RTI.

1. Επιστρέφει στη διεύθυνση στην κορυφή της στοίβας του PC.
2. Γίνεται pop της στοίβας του PC.
3. Γίνεται pop της στοίβας αν χρειάζεται.
4. Καθαρίζονται τα bits στον interrupt latch καταχωρητή και στον IMASKP.

Απόδοση της διακοπής στον SHARC

Τρεις βαθμίδες απόκρισης:

- 1 κύκλος: συγχρονισμός και σύρσιμο,
- 1 κύκλος: αναγνώριση,
- 2 κύκλοι: διακλάδωση στο διάνυσμα.

Συνολική επιβάρυνση: 3 κύκλοι.

Τα διανύσματα των διακοπών σε μικροεπεξεργαστές έχουν επιβάρυνση 6 κύκλων.

Λειτουργία τύπου υπερχρήστη

- Μπορεί να θέλει να παράσχει προστατευτικά σύνορα ανάμεσα σε προγράμματα.
  - Αποφυγή φθοράς της μνήμης.
- Χρειάζεται λειτουργία τύπου υπερχρήστη για το χειρισμό διαφόρων προγραμμάτων.
- Ο SHARC δεν παρέχει λειτουργία τύπου υπερχρήστη.

Λειτουργία τύπου υπερχρήστη στον ARM

- Χρησιμοποιείται η εντολή SWI για είσοδο σε λειτουργία τύπου υπερχρήστη, όμοια με υπορουτίνα:

```
SWI CODE_1
```

- Τίθεται ο PC σε 0x08.
- Το όρισμα του SWI περνά σε όρισμα λειτουργίας τύπου υπερχρήστη.
- Σώζεται ο CPSR στον SPSR.

Εξαίρεση (exception)

- Εξαίρεση: εσωτερικά ανιχνεύσιμο σφάλμα.
- Οι εξαιρέσεις είναι σύγχρονες με τις εντολές αλλά απρόβλεπτες.
- Ο μηχανισμός εξαιρέσεων χτίζεται πάνω από το μηχανισμό διακοπής.
- Οι εξαιρέσεις συνήθως διανυσματοποιούνται και τους ορίζεται προτεραιότητα.

Παγίδα (trap)

- Παγίδα (διακοπή λογισμικού): μια εξαίρεση που δημιουργείται από μια εντολή.
  - Γίνεται κλήση λειτουργίας τύπου υπερχρήστη.
- Ο ARM χρησιμοποιεί εντολές SWI για παγίδες.
- Ο SHARC προσφέρει τρία επίπεδα διακοπής λογισμικού.
  - Γίνεται κλήση από κάνοντας set τα bits του καταχωρητή IRPTL.

Συνεπεξεργαστής

- Συνεπεξεργαστής: προστιθέμενη λειτουργική μονάδα που καλείται με εντολή.
  - Μονάδες κινητής υποδιαστολής συνήθως δομούνται σαν συνεπεξεργαστές.
- Ο ARM μέχρι και 16 συνεπεξεργαστές που επιλέγονται από το σχεδιαστή.
  - Ο συνεπεξεργαστής κινητής υποδιαστολής χρησιμοποιεί τις μονάδες 1 και 2.