Ενσωματωμένα Συστήματα

Ενότητα 7: Κεντρικές Μονάδες Επεξεργασίας.

Δρ. Μηνάς Δασυγένης
mdasyg@ieee.org
Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών
http://arch.icte.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σκοπός ενότητας

- Η κατανόηση των τεχνικών Εισόδου Εξόδου που χρησιμοποιούνται στα ενσωματωμένα συστήματα.

- Η κατανόηση των διακοπών και των εξαιρέσεων στα ενσωματωμένα συστήματα με παραδείγματα στον επεξεργαστή ARM.

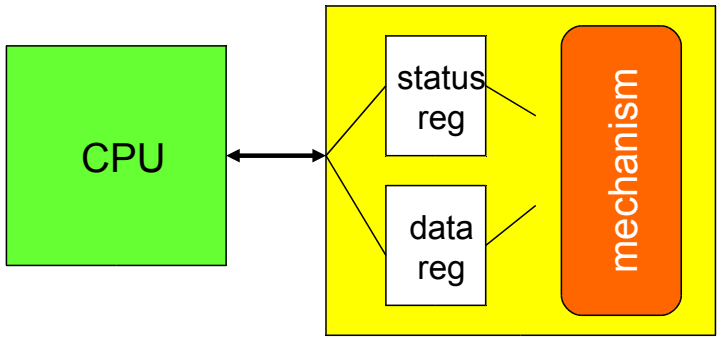
Δομή της παρουσίασης

- Μηχανισμοί Εισόδου και Εξόδου.
- Κατάσταση λειτουργίας επιβλέποντος *(supervisor)*, εξαιρέσεις και παγίδες.
- Συνεπεξεργαστές.

Όλα τα ανωτέρω είναι στοιχεία απαραίτητα για τη διασύνδεση της CPU με άλλα στοιχεία του συστήματος.

Συσκευές Εισόδου-Εξόδου (I/O)

- Οι συσκευές I/O συνήθως έχουν κάποια αναλογικά ή μη ψηφιακά συστατικά *(π.χ. η κεφαλή ανάγνωσης/εγγραφής ενός σκληρού δίσκου).*
- Η ψηφιακή λογική σε κάθε I/O είναι παρόμοια.
- Τυπική διεπαφή I/O με CPU:



Συσκευές Εισόδου-Εξόδου Καταχωρητές

- Η επικοινωνία *(CPU - I/O)* γίνεται με τους καταχωρητές της συσκευής.
- Οι συσκευές έχουν αρκετούς καταχωρητές:
  - **Καταχωρητές Δεδομένων** *(data registers)* κρατούν τιμές που αντιμετωπίζονται ως δεδομένα από τη συσκευή *(π.χ. αυτά που γράφονται στο δίσκο).*
  - **Καταχωρητές Κατάστασης** *(status registers)* παρέχουν πληροφορίες σχετικά με τη λειτουργία της συσκευής *(π.χ. αν τα δεδομένα θα γραφούν ή θα διαβαστούν).*
  - Επίσης, οι καταχωρητές χωρίζονται σε:
    - ✓ εγγραφής/ανάγνωσης (**RW**).
    - ✓ μόνο ανάγνωσης (**RO**).

Παράδειγμα εφαρμογής 8251 UART

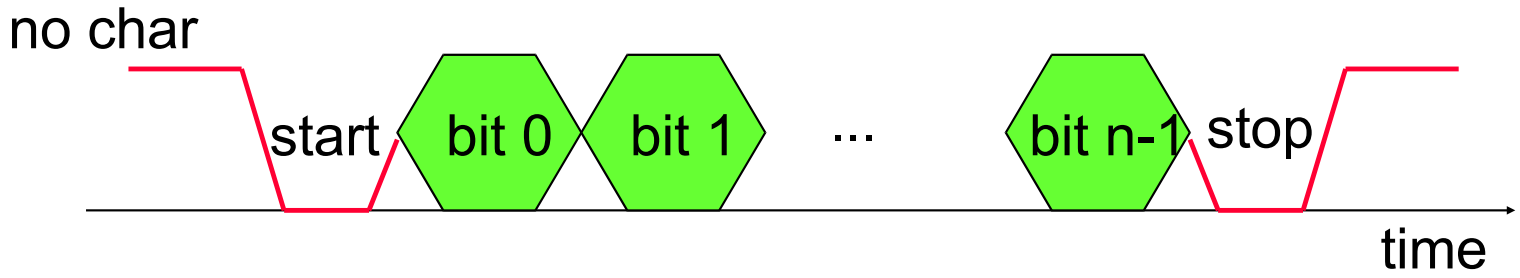
Application: 8251 UART

- Ασύγχρονος πομποδέκτης γενικής χρήσης *(σειριακή επικοινωνία).*
- Οι σύγχρονοι υπολογιστές έχουν το chip 16550.

- **Universal asynchronous receiver transmitter** *(UART)* : παρέχει σειριακές επικοινωνίες.
- Το 8251 αρχικά ήταν αυτόνομο IC.
- Σήμερα, οι λειτουργίες του 8251 εντάσσονται σε ένα μεγαλύτερο ολοκληρωμένο κύκλωμα *(με περισσότερες λειτουργίες)*.
- Χρησιμοποιείται ακόμη η βασική προγραμματιστική διασύνδεση.
- Επιτρέπει τον προγραμματισμό πολλαπλών παραμέτρων επικοινωνίας.

Η σειριακή επικοινωνία

- Οι χαρακτήρες μεταδίδονται σειριακά ως ρεύματα χαρακτήρων *(με start/stop bit).*
- Η ταχύτητα μετάδοσης *(baud rate)* είναι σταθερή.
- Οι χαρακτήρες μεταδίδονται ξεχωριστά:



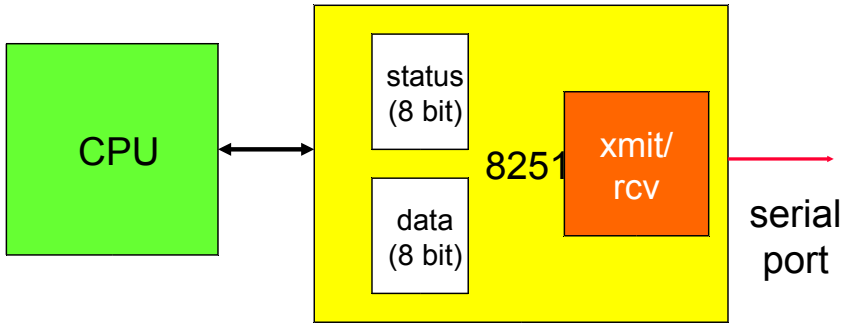
- Υπάρχει bit έναρξης/λήξης και bit σταματήματος.
- Η περίοδος ενός bit είναι το αντίστροφο της ταχύτητας μετάδοσης.

Παράμετροι Σειριακής Επικοινωνίας

- Πριν τη μετάδοση δεδομένων η CPU πρέπει να θέσει τους καταχωρητές κατάστασης λειτουργίας *(mode registers)*, ώστε να ανταποκρίνονται στα χαρακτηριστικά της γραμμής δεδομένων:
  - **(ταχύτητα)** Baud (bit) rate.
  - **(αριθμός bit/ χαρακτήρα)** Number of bits per character.
  - **(ύπαρξη ισοτιμίας)** Parity/no parity.
  - **(είδος ισοτιμίας)** Even/odd parity.
  - **(μήκος bit σταματήματος)** Length of stop bit *(1, 1.5, 2 bits).*

Η διεπαφή του 8251

- Το UART περιλαμβάνει ένα καταχωρητή 8bit ανάμεσα στο CPU και στο δίαυλο, για την αποθήκευση των δεδομένων.
- Υπάρχει και ο καταχωρητής κατάστασης, που χρησιμοποιείται αν η συσκευή έχει λάβει δεδομένα *(Receiver Ready)* ή μπορεί να στείλει δεδομένα *(Transmitter Ready).*



Θεμελιώδη στοιχεία εισόδου και εξόδου

- Προκειμένου να χρησιμοποιηθούν οι συσκευές εισόδου εξόδου, θα πρέπει να υπάρχει αντίστοιχη υποστήριξη από το ISA.
- Υπάρχουν διάφορες τεχνικές για I/O *(η κάθε μια έχει πλεονεκτήματα και μειονεκτήματα).*
  - Προγραμματιζόμενη I/O.
  - Χαρτογραφημένη I/O.
  - I/O με DMA *(δε θα αναλυθεί σε αυτή τη διάλεξη).*

Προγραμματιστική Υποστήριξη I/O

- Οι I/O υποστηρίζονται από δυο ομάδες εντολών :
  - Εντολές I/O ειδικού σκοπού *(programmable I/O).*
  - Εντολές μεταφοράς μέσω χαρτογράφησης μνήμης *(memory mapped).*
- Η αρχιτεκτονική Intel x86 υποστηρίζει τις εντολές **in**, **out**. Επίσης υποστηρίζεται και χαρτογράφηση μνήμης με την εντολή **mov** *(π.χ. εγγραφή στην οθόνη).*
  Οι άλλες CPU χρησιμοποιούν κυρίως I/O με απεικόνιση στη μνήμη *(ο πιο συνηθισμένος τρόπος).*
- Οι εξειδικευμένες εντολές I/O δεν αποκλείουν την I/O μέσω χαρτογράφησης μνήμης.

I/O στον ARM με χαρτογράφηση μνήμης

- Ορίζεται ένα συμβολικό όνομα για τη θέση μνήμης της συσκευής:

```
DEV1 EQU 0x1000
```

- Κώδικας για εγγραφή και ανάγνωση:

```
LDR r1,#DEV1      ;set up device adrs
LDR r0,[r1]       ;read DEV1
LDR r0,#8         ;set up value to write
STR r0,[r1]       ;write value to device
```

I/O με γλώσσα προγραμματισμού υψηλού επιπέδου

- Δεν υπάρχει εύκολα πρόσβαση σε απόλυτη διεύθυνση. Ο compiler χρησιμοποιεί σχετικές διευθύνσεις.
- Μόνο με τη χρήση δεικτών μπορούμε να έχουμε πρόσβαση σε συγκεκριμένη διεύθυνση μνήμης.
- Αν δεν υπάρχει λειτουργικό σύστημα *(οπότε και δε θα μεταφραστεί η διεύθυνση σύμφωνα με τον πίνακα σελίδων)*, τότε μπορούμε να χρησιμοποιήσουμε τις συναρτήσεις **peek()**, **poke()**.

Οι συναρτήσεις peek(), poke()

Τυπικοί τρόποι πρόσβασης σε I/O:

- Ανάγνωση μέσω pointer

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

Παράδειγμα Χρήσης:

```
#define DEV1 0x1000
dev_status=peek(DEV1);
```

- Εγγραφή μέσω pointer

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

Παράδειγμα Χρήσης:

```
#define DEV1 0x1000
poke(dev1,8);
```

I/O με αναμονή λόγω απασχόλησης

- Ο πιο βασικός & αναποτελεσματικός τρόπος.
- Μεγάλη σπατάλη κύκλων.
- Βασίζεται στο γεγονός ότι οι συσκευές λειτουργούν με πολύ μικρότερη συχνότητα από ότι ο επεξεργαστής.
- Συνεχώς ελέγχουμε τη διαθεσιμότητα της συσκευής ή αν έχει ολοκληρώσει το έργο που της είχε ανατεθεί για να αναλάβει τα επόμενα δεδομένα I/O.
- Θα είναι λάθος αν η συσκευή είναι απασχολημένη με I/O και τις στείλουμε και άλλα δεδομένα για I/O.
- Ονομάζεται και συχνά περιόδευση *(polling).*

Προγραμματισμός I/O με αναμονή λόγω απασχόλησης (3-3)

- Παράδειγμα εγγραφής χαρακτήρων σειριακά.
- Η συσκευή έχει 2 καταχωρητές: κατάστασης *(1Byte)*, δεδομένων *(1Byte).*
- Αν ο καταχωρητής κατάστασης έχει τιμή 1 τότε η συσκευή είναι απασχολημένη.

```
#define OUT_CHAR 0x1000
#define OUT_STATUS 0x1001
char *mystring = "Hello, world."
char *current_char;
current_char = mystring;
while (*current_char != '\0') {
poke(OUT_CHAR,*current_char);
while (peek(OUT_STATUS) != 0); ← Busy Wait
current_char++; }
```

Ταυτόχρονη είσοδο έξοδο με αναμονή λόγου απασχόλησης (3-4)

```
#define IN_DATA 0x1000
#define IN_STATUS 0x1001
#define OUT_CHAR 0x1100
#define OUT_STATUS 0x1101

...
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);
        }
```

- Μόλις διαβαστεί κάποιος χαρακτήρας από την είσοδο, στέλνεται στην έξοδο.
- Χρησιμοποιούνται καταχωρητές κατάστασης.

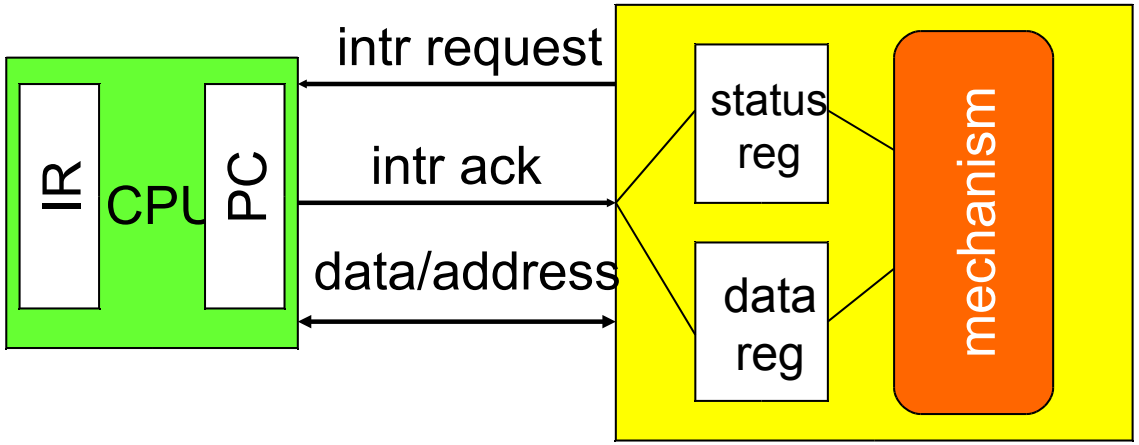
Interrupt I/O

- **Η I/O με αναμονή λόγω απασχόλησης είναι εξαιρετικά αναποτελεσματική.**
- Η CPU δεν κάνει τίποτα άλλο παρά να εξετάζει την κατάσταση της συσκευής.
- Δεν είναι εύκολο να γίνει παράλληλα άλλη εργασία I/O.
  - Υπολογισμό για τα επόμενα δεδομένα I/O.
  - Έλεγχο άλλων συσκευών εισόδου/εξόδου.
- Υπάρχει όμως λύση: I/O με διακοπές.

I/O με διακοπές: ένας αρκετά χρήσιμος μηχανισμός I/O

- Επιτρέπει τις συσκευές να σηματοδοτούν τη CPU, ώστε να αλλάξει η τυπική ροή εκτέλεσης κώδικα, και να εκτελέσει ειδικό κώδικα *“ρουτίνας χειρισμού διακοπής (interrupt handler routine)”.*
- Ο κώδικας ονομάζεται και οδηγός συσκευής.
- Επειδή τροποποιείται το PC,αποθηκεύεται η παλαιά του τιμή.
- Η υποστήριξη διακοπών απαιτεί αρχιτεκτονικές τροποποιήσεις στη ΚΜΕ και πρέπει να αναλυθεί πότε θα γίνει η διακοπή *(αμέσως ή μετά την ολοκλήρωση της εκτέλεσης μιας εντολής).*

Ο μηχανισμός διακοπής



2 βήματα σηματοδοσίας

- 1/2 Η συσκευή Ε/Ε ενεργοποιεί το σήμα αίτησης διακοπής INT.
- 2/2 Η CPU ενεργοποιεί το σήμα επιβεβαίωσης διακοπής όταν αρχίζει να εξυπηρετεί τη διακοπή *(η επιβεβαίωση μπορεί να γίνει μετά από λίγο, αν π.χ. υπάρχει άλλη εξυπηρέτηση διακοπής).*

Μπορεί να υποστηρίζονται και προτεραιότητες.

Η υλοποίηση των διακοπών

- Βασίζεται στο μηχανισμό κλήσεων συναρτήσεων.
- ..αλλά δεν καλείται άμεσα από το πρόγραμμα.
- Η διακοπή αναγκάζει την επόμενη εντολή να είναι μια κλήση συνάρτησης σε μια προδιαγεγραμμένη διεύθυνση.
- Διευκολύνει τις θεματικές εναλλαγές, μεταξύ ενός υπολογισμού στο προσκήνιο και συσκευών I/O.
- Οι προτεραιότητες επιτρέπουν την ΚΜΕ να επιλέγει τι πιο σημαντικές διακοπές *(π.χ. δίσκος)*, από τις όχι τόσο σημαντικές διακοπές *(π.χ. πληκτρολόγιο).*
- Το πρόγραμμα που εκτελείται όταν δεν εξυπηρετείται καμία διακοπή, ονομάζεται **πρόγραμμα στο προσκήνιο** *(foreground program).*

Παράδειγμα I/O με διακοπές: ρουτίνες εξυπηρέτησης

```
/* …global achar,gotchar */
void ISR_input_handler() {
        achar = peek(IN_DATA);
        gotchar = TRUE;
        poke(IN_STATUS,0);
}

void ISR_output_handler() { }
```

Η gotchar είναι καθολική μεταβλητή για σηματοδοσία

Υλοποίηση συναρτήσεων χειρισμού διακοπών στη C

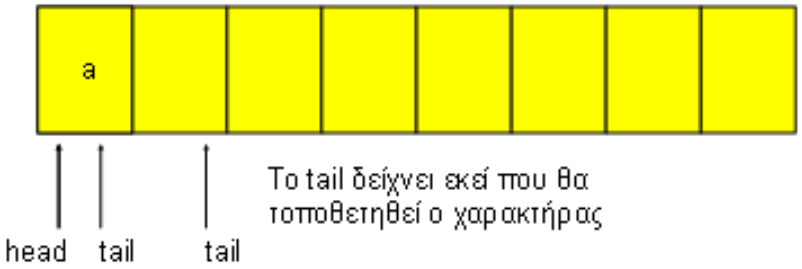
Παράδειγμα I/O με διακοπές: κυρίως πρόγραμμα

```
main() {
        while (TRUE) {
                if (gotchar) {

        poke(OUT_DATA,achar);

        poke(OUT_STATUS,1);
                        gotchar = FALSE;
                        }
                }
        }
```

Αν και έχει βελτιωθεί, εντούτοις ακόμη δεν αφήνει το πρόγραμμα στο προσκήνιο να κάνει χρήσιμη δουλειά.

Παράδειγμα I/O με διακοπές και προσωρινή μνήμη

- Πιο πολύπλοκη υλοποίηση, αλλά πιο αποτελεσματική.
- Ουρά χαρακτήρων:



Επιτρέπει τη λειτουργία Ε/Ε σε διαφορετικούς ρυθμούς (διαφορετικό ρυθμό ανάγνωσης,εγγραφής).

Καθολικές Μεταβλητές:

- **io_buf (buffer)**
- **buf_start , buf_end** *(πρώτο και τελευταίο χαρακτήρα)*
- **error** *(αν 0 υπερχείλιση)*

Συνάρτηση χειρισμού διακοπής εισόδου με προσωρινή μνήμη

```
void ISR_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); }
}
```

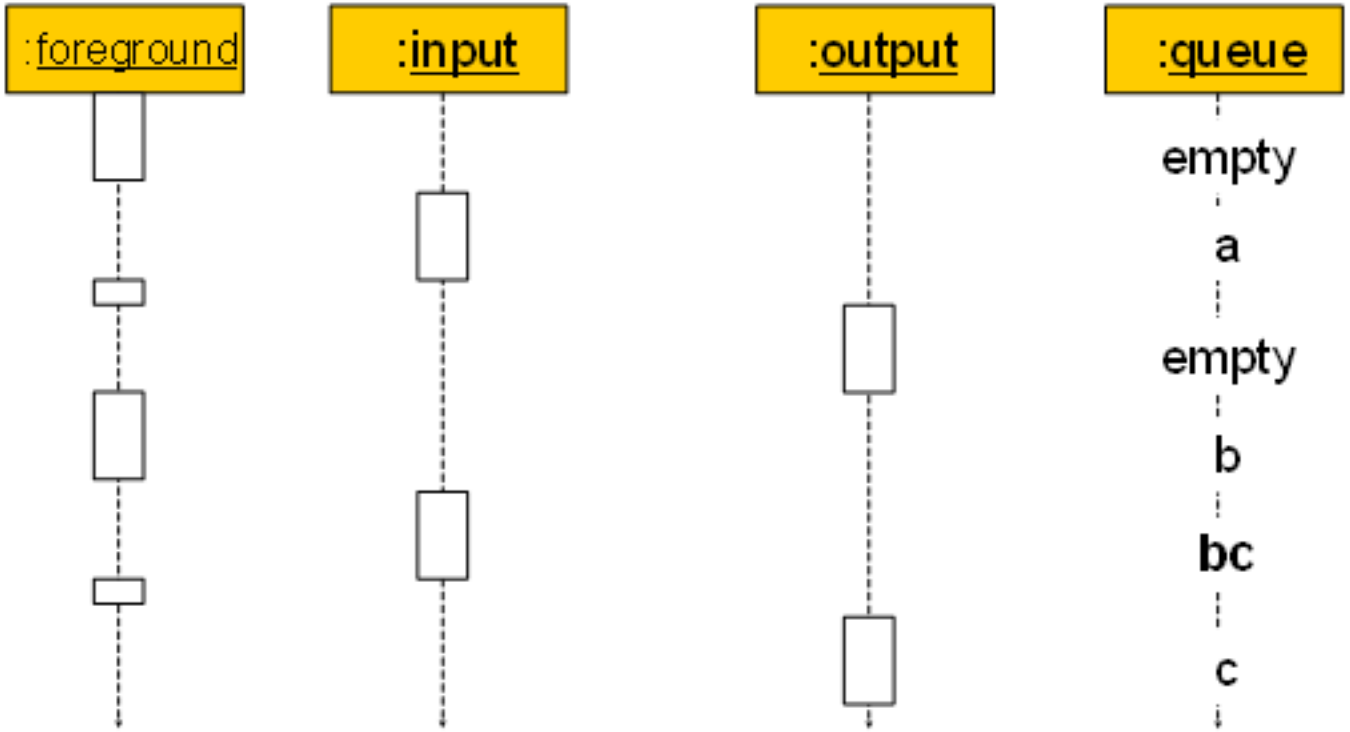
Συνάρτηση χειρισμού διακοπής εξόδου με προσωρινή μνήμη

```
void ISR_output_handler()
{
if (!empty_buffer())
poke(OUT_DATA,remove_char
());
poke(OUT_STATUS,1);
}
```

Η συνάρτηση αυτή χρησιμοποιείται μόνο για πρώτη φορά κατά την εκκίνηση του προγράμματος
(αν υπάρχουν χαρακτήρες για αποστολή, για να αδειάσει το buffer).

Σε κάθε επόμενη χρήση, η συνάρτηση εισόδου στέλνει τους χαρακτήρες στην έξοδο.

Διάγραμμα ακολουθίας I/O



Το πρόγραμμα στο προσκήνιο είναι ανεξάρτητο.

Αυξάνεται ο ταυτοχρονισμός/παραλληλισμός.

Σφάλματα σε χειριστές διακοπών

- Χειρισμός διακοπής χωρίς επαναφορά όλων των καταχωρητών:
  - Το πρόγραμμα στο προσκήνιο θα έχει παράξενη συμπεριφορά.
  - Τα bugs είναι πολύ δύσκολο να επαναληφθούν, αυτό οφείλεται στο πότε θα συμβεί η διακοπή.

Παράδειγμα αποσφαλμάτωσης κώδικα διακοπής

- Έστω η συνάρτηση χειρισμού διακοπής δεν είναι “διαφανής”.
- Τότε κάποιοι καταχωρητές που τροποποιούνται δεν επαναφέρονται στις αρχικές τους τιμές.
- Αναλόγως πότε θα κληθεί η συνάρτηση αυτή, τότε ενδέχεται να προκαλέσει πρόβλημα ή να μην εμφανιστεί η προβληματική κατάσταση.
- Π.χ το πρόγραμμα στο προσκήνιο:

```
for (i=0;i<M;i++){ y[i]=b[i]; for (j=0;j<N;j++)
y[i]=y[i]+A[i,j]*x[j];}
```

- Τι θα συμβεί αν τροποποιηθεί το j από το χειριστή; Εξαρτάται από το πότε θα συμβεί αυτό
  (μια διακοπή στην αρχή του βρόχου, θα δώσει διαφορετικό αποτέλεσμα από τη διακοπή στο τέλος του βρόχου).

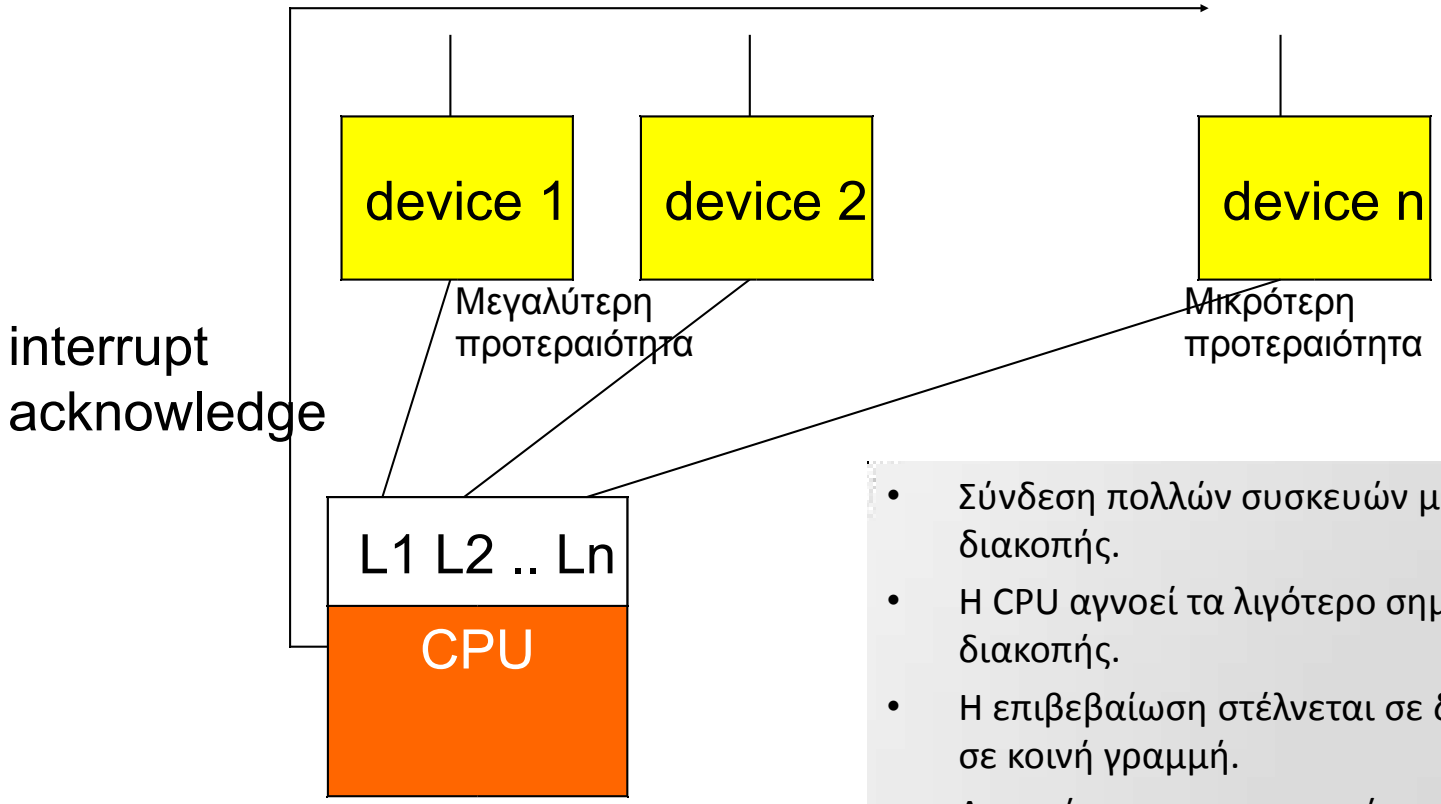
Οι διακοπές απαιτούν σωρό

- Η CPU ελέγχει για διακοπές σε κάθε εντολή assembly *(απαιτείται γρήγορη απόκριση).*
- Ο μηχανισμός διακοπών χτίζεται πάνω στις υπορουτίνες *(στοίβα, διεύθυνση επιστροφής, κτλ).*
- Η στοίβα μπορεί να είναι ίδια με τις διεργασίες ή “ειδικού τύπου”.

Προτεραιότητες και Διανύσματα

- Δυο μηχανισμοί που επιτρέπουν τις διακοπές να γίνουν πιο συγκεκριμένες
  (γενίκευση για περισσότερες συσκευές):
  - **Προτεραιότητες**
    (καθορίζουν ποια διακοπή εξυπηρετείται πρώτα).
  - **Διανύσματα**
    (καθορίζουν ποιος κώδικας καλείται για κάθε διαφορετική διακοπή
    {ύπαρξη πολλαπλών interrupt lines}).
- Είναι ορθογώνιοι μηχανισμοί. Υποστηρίζονται συνήθως και οι δυο.

Διακοπές με προτεραιότητα



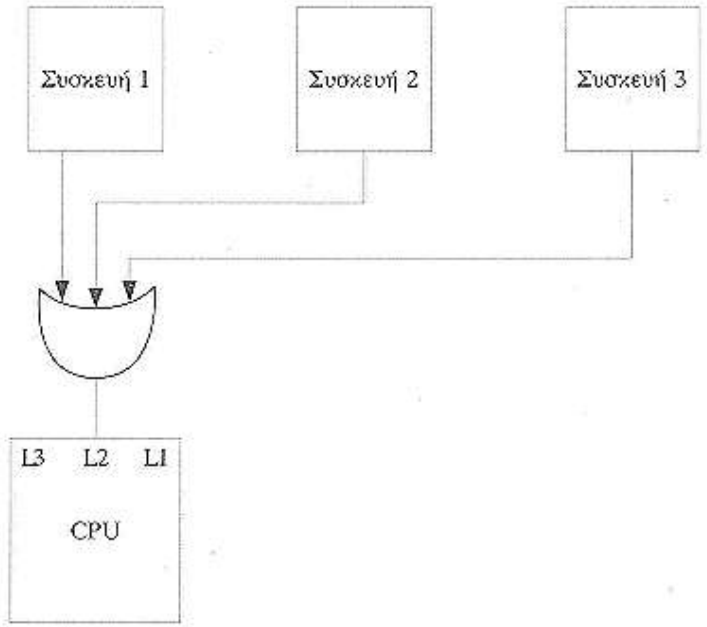
- Σύνδεση πολλών συσκευών με τη γραμμή διακοπής.
- Η CPU αγνοεί τα λιγότερο σημαντικά αιτήματα διακοπής.
- Η επιβεβαίωση στέλνεται σε δυαδική μορφή σε κοινή γραμμή.
- Δυνατότητα για τροποποίηση προτεραιότητας με σύνδεση περιφερειακού σε διαφορετική γραμμή.

Διακοπές με προτεραιότητα: Δυνατότητα εφαρμογής μάσκας

- **Masking** *(εφαρμογή μάσκας)*: μια διακοπή με χαμηλότερη προτεραιότητα από αυτήν που εξυπηρετείται, δεν αναγνωρίζεται έως ότου να ολοκληρωθεί η τρέχουσα προτεραιότητα.
- **Non-maskable interrupt** *(NMI – μη αποκρύψιμη διακοπή)*: Η διακοπή με τη μεγαλύτερη προτεραιότητα δεν έχει ποτέ μάσκα, και εξυπηρετείται άμεσα πάντα.
  - Χρησιμοποιείται για διακοπές υψίστης προτεραιότητας *(π.χ. Απώλεια ισχύος μπαταρίας στο laptop, ώστε να αποθηκευτούν τα δεδομένα στο δίσκο).*

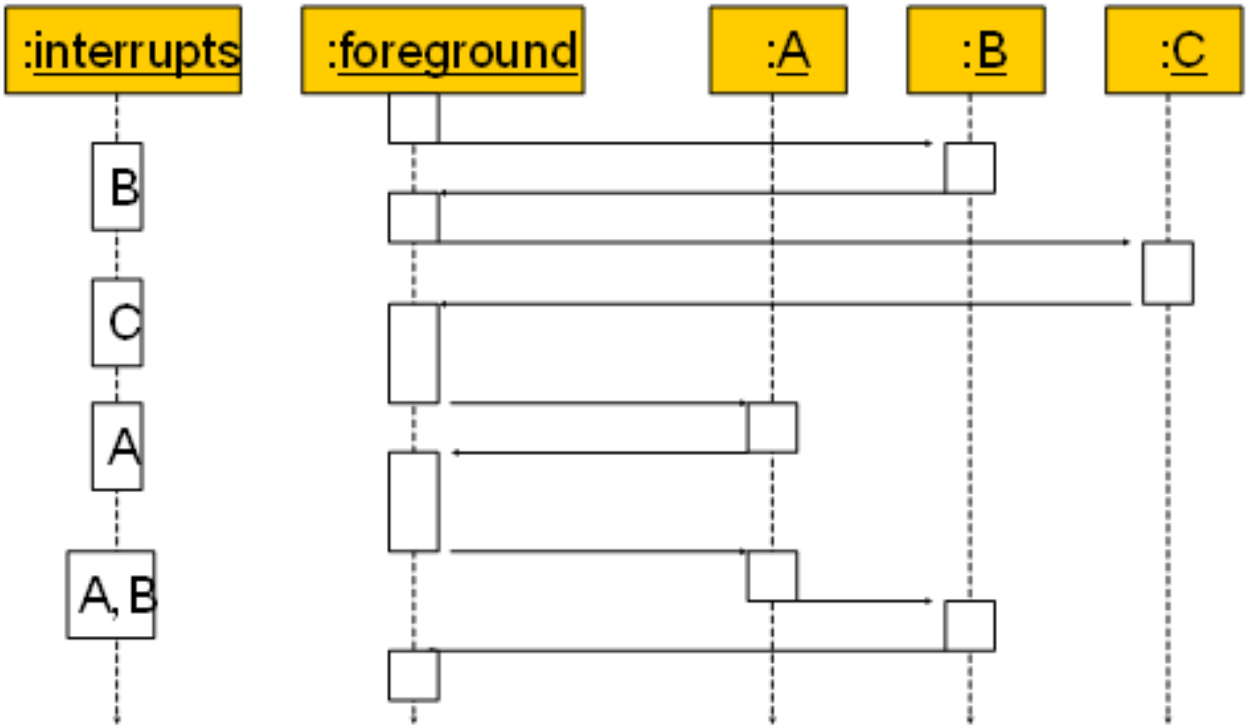
    ✓ Υπάρχει ειδική εντολή επιστροφής από interrupt (*στη x86 είναι η IRET)*, και αυτό οφείλεται στην επαναφορά του καταχωρητή επιπέδου προτεραιότητας *(κάτι που δε θα συνέβαινε αν υπήρχε η τυπική επιστροφή από ρουτίνα).*

Συνδυασμός περιόδευσης για κοινή χρήση διακοπής



- Οι γραμμές διακοπής είναι περιορισμένες *(π.χ. ο 8086 έχει 5 γραμμές, pentium 256).*
- Δεν είναι απαραίτητο κάθε συσκευή να συνδέεται σε ξεχωριστή γραμμή IRQ.
- Συσκευές με ίδια προτεραιότητα μπορούν να συνδεθούν με μικρή ποσότητα λογικής.
- Ο χειριστής διακοπής θα κάνει polling για να βρει τη συσκευή που προκάλεσε τη διακοπή.

Παράδειγμα προτεραιοτήτων



Α: υψηλότερη προτεραιότητα

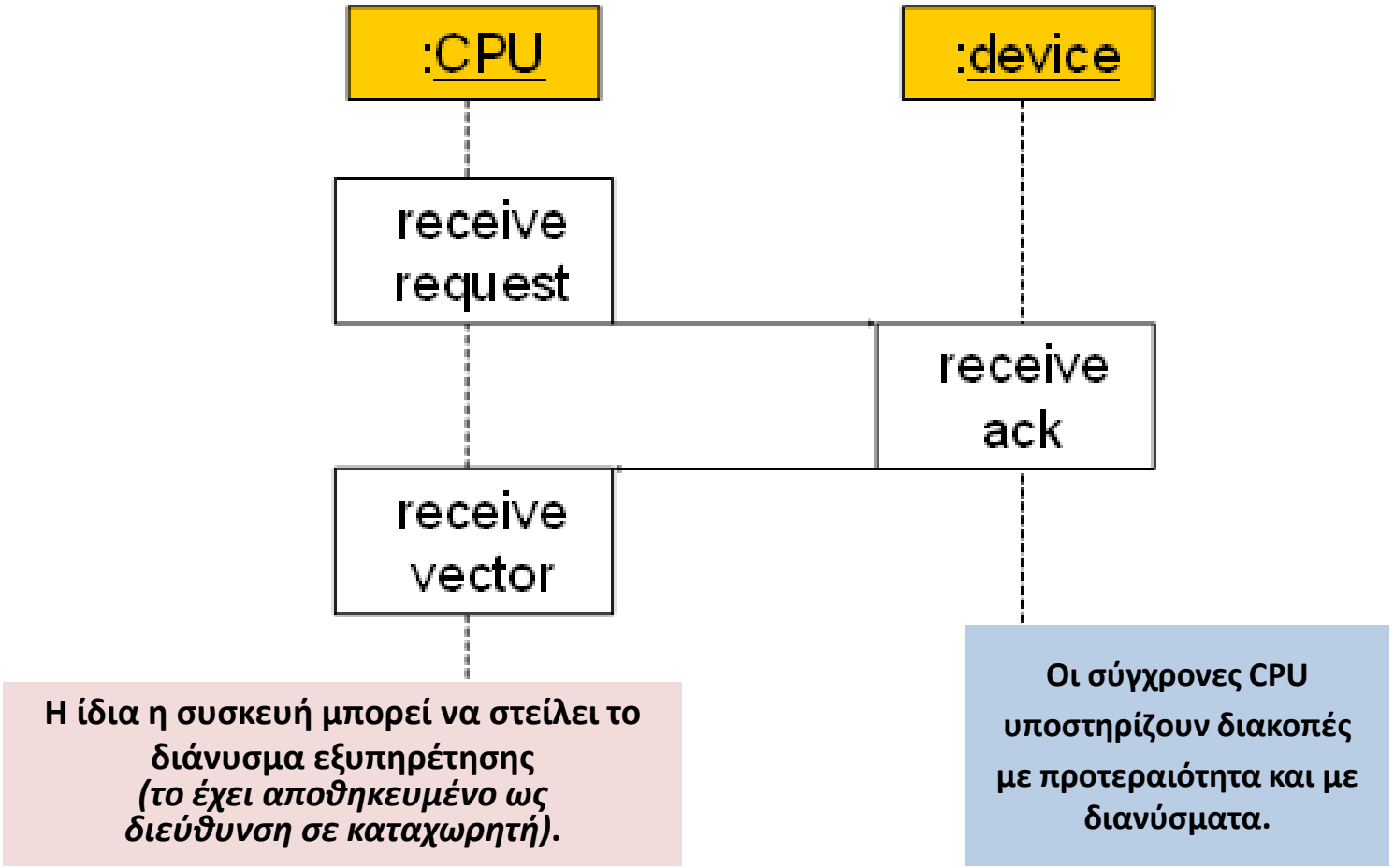
- Πρέπει να λαμβάνεται υπόψιν ο χειρότερος συνδυασμός διακοπών.

Διανύσματα Διακοπής

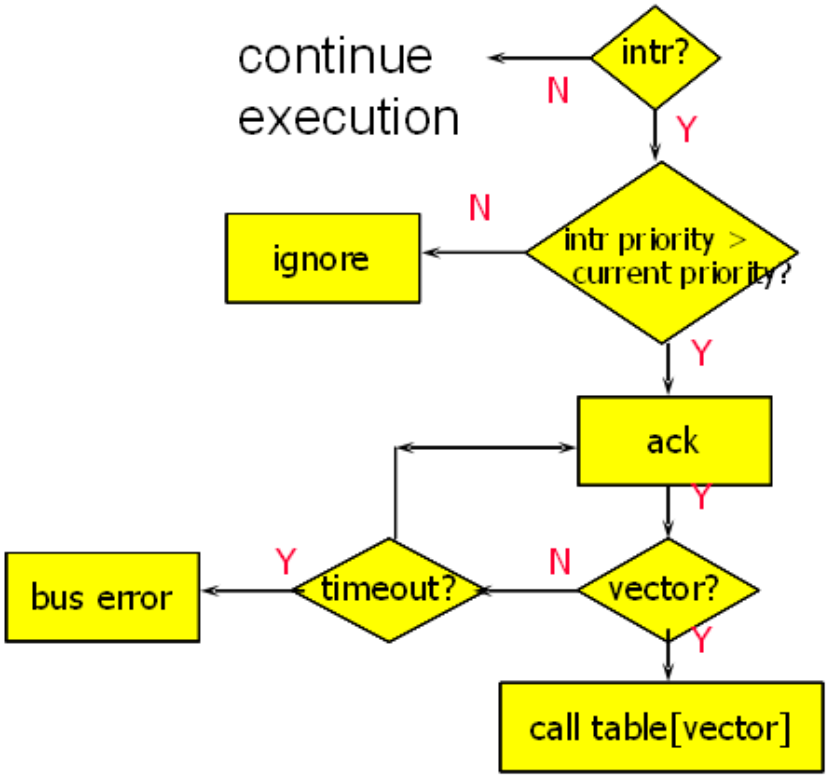
- Επιτρέπει διαφορετικές διακοπές να έχουν διαφορετική συνάρτηση διακοπής.
- Μπορεί το διάνυσμα να σταλεί από τη συσκευή, ή μπορεί να βρίσκεται ήδη σε ειδική διεύθυνση μνήμης
  (π.χ. η διακοπή 0, αντιστοιχεί στη διεύθυνση μνήμης 0, που συνήθως έχει ένα JMP προς την αντίστοιχη συνάρτηση).
- Πίνακας διανυσμάτων διακοπής

Μας επιτρέπει να καθορίζουμε το χειριστή διακοπών κάθε IRQ.

Αποστολή διανύσματος διακοπής



Γενικός μηχανισμός χειρισμού interrupt



Assume priority selection is handled before this point.

Κατά το interrupt, βήματα εκτελούνται από:

- Συσκευή.
- CPU.
- Λογισμικό.

Ακολουθία διακοπής

- Η CPU επιβεβαιώνει τη διακοπή.
- Η συσκευή στέλνει το διάνυσμα.
- Η CPU καλεί τη συνάρτηση χειρισμού.
- Εκτελείται το λογισμικό για το χειρισμό της διακοπής.
- Η CPU επαναφέρει πλήρως την κατάσταση στο πρόγραμμα στο παρασκήνιο.

Υπάρχει επιβάρυνση στο χρόνο εκτέλεσης που σχετίζεται με το μηχανισμό διακοπών.....

Πηγές επιβάρυνσης διακοπών

- Χρόνος εκτέλεσης κώδικα χειρισμού διακοπής.
- Επιβάρυνση χειρισμού διακοπής από ΚΜΕ *(π.χ. επιβεβαίωση).*
- Καθυστέρηση αποθήκευση καταχωρητών στο σωρό.
- Καθυστερήσεις διασωλήνωσης *(ποινή διακλάδωσης).*
- Ποινές κρυφής μνήμη.

Ο χρόνος απόκρισης διακοπών εξαρτάται από το CPU
(π.χ. στο τι δεδομένα αποθηκεύουν σε μια διακοπή-πρέπει να είναι ελάχιστα).

Διακοπές στον ARM

- Ο ARM7 υποστηρίζει δύο τύπους διακοπής:
  - Γρήγορες αιτήσεις διακοπής (**FIQs**) *(high priority).*
  - Αιτήσεις διακοπής (**IRQs**).
  - Η διακοπή πίνακα ξεκινά στη θέση 0.

Διαδικασίες διακοπών στον ARM7

- Ενέργειες CPU:
  - Αποθηκεύει PC.
  - Αντιγράφει το CPSR στο SPSR.
  - Εξαναγκάζει τη ρύθμιση των bits μέσα στο CPSR για να καταγράψουν τη διακοπή.
  - Ρύθμιση του νέου PC.
- Ευθύνες Χειριστή :
  - Επαναφορά σωστού PC.
  - Επαναφορά CPSR από SPSR.
  - Καθαρισμός των bit διακοπής, απενεργοποίηση σημαιών.

Καθυστέρηση διακοπής ARM7

- Στη χειρότερη περίπτωση καθυστέρησης, η ανταπόκριση είναι 27 κύκλοι:
  - Δύο κύκλοι να συγχρονιστεί το εξωτερικό αίτημα.
  - Έως 20 κύκλοι για να ολοκληρωθεί η τρέχουσα εντολή.
  - Τρεις κύκλοι για την ακύρωση των δεδομένων.
  - Δύο κύκλοι να εισέλθει σε κατάσταση διακοπής χειρισμού.

Ρυθμός επιβλέποντος

- Παροχή ελέγχου και προστασίας ανάμεσα σε πολλαπλές διεργασίες στον ίδιο υπολογιστή
- Π.χ. Αποφυγή πρόσβασης σε περιοχές μνήμης.
- Απαιτείται ο ρυθμός επιβλέποντος *(supervisor mode).*
- Απαιτείται για ένα λειτουργικό σύστημα με πολλαπλές διεργασίες.
- Δεν έχουν όλοι οι επεξεργαστές αυτόν το ρυθμό *(π.χ. δεν το έχει ο TI DSP Sharc ή ο 8086).*

✓ **Οι έλεγχοι από υλικό εξασφαλίζουν ένα επιπρόσθετο επίπεδο ασφάλειας.**

- Επιτρέπονται λειτουργίες που δεν υποστηρίζονται στην κατάσταση χρήστη. Π.χ. Έλεγχος διαχείρισης μνήμης, χρήση dma, I/O.

Η κατάσταση επιβλέποντος

- Υποστηρίζεται από τη CPU.
- Τα κανονικά προγράμματα εκτελούνται σε κατάσταση χρήστη.
- Η κατάσταση επιβλέποντος έχει προνόμια και μπορεί να εκτελέσει περισσότερες εντολές *(π.χ. διαχείριση μνήμης, I/O,..).*
- Συνήθως τμήματα του ΛΣ εκτελούνται σε αυτήν την κατάσταση.

ARM supervisor mode

- Χρήση της εντολής SWI :

```
SWI CODE_1
```

- Θέτει PC στο 0x08.
- Οι παράμετροι στο SWI χρησιμοποιούνται στο supervisor mode κώδικα
  (αίτηση για συγκεκριμένη υπηρεσία).
- Τα 5 χαμηλότερα bit του CPSR γίνονται '1'.
- Αποθηκεύεται ο CPSR στο SPSR.
- Για επιστροφή από το supervisor mode:
- R14→R15, SPSR→CPSR

Εξαίρεση

- **Εξαίρεση** *(exception)* : εσωτερικά ανιχνευμένο σφάλμα
  (π.χ. διαίρεση με το 0, unalligned memory access).
- Οι εξαιρέσεις είναι σύγχρονες με τις εντολές, αλλά μη προβλέψιμες.
- Παρόμοια με interrupt, γιατί προκαλούν αλλαγή ροής προγράμματος.
- Διαφορετικά με interrupt, γιατί παράγονται εσωτερικά.
- Χρησιμοποιείται ο μηχανισμός των διακοπών για την υλοποίηση.
- Υποστηρίζουν προτεραιότητες και διανύσματα.

Παγίδες

- **Παγίδα (Trap)** *(διακοπή λογισμικού)*: μια διακοπή που παράγεται από μια εντολή.
  - Π.χ. Κλήση εντολής για μετάβαση σε κατάσταση επιβλέποντος *(επειδή απαιτείται έλεγχος, λόγω αύξησης προνομίων).*
- Το ARM χρησιμοποιεί SWI εντολές για παγίδες.

Συνεπεξεργαστές

- **Συνεπεξεργαστής** *(Co-processor)*
  - Αυτόνομη και συνδεδεμένη μονάδα και με τη CPU που παρέχει επιπρόσθετες λειτουργίες που ενεργοποιούνται με κατάλληλες εντολές.
    - ✓ Π.χ. Μονάδες πράξεων πραγματικών αριθμών.
  - Δεσμεύονται ορισμένοι opcodes για τις λειτουργίες του συνεπεξεργαστή.
  - Αν δεν υπάρχει ο συνεπεξεργαστής, τότε η χρήση co-op που αντιστοιχεί σε αυτόν, δημιουργεί παγίδα *(εσφαλμένη εντολή)* .... μπορεί να χρησιμοποιηθεί εξομοίωση σε αυτή την περίπτωση.
- Είναι στενά συνδεδεμένοι *(tightly coupled),* δηλαδή επιτρέπεται αμφίδρομη πρόσβαση στους καταχωρητές.
- Μπορεί η CPU να λειτουργήσει ως superscalar, δηλαδή να συνεχίσει την εκτέλεση άλλης εντολής *(παράλληλα).*

Συνεπεξεργαστές στον ARM

- Ο ARM επιτρέπει τη σύνδεση έως 16 συνεπεξεργαστών.
- Οι συνεπεξεργαστές εκτελούν λειτουργίες φόρτωσης/αποθήκευσης στους δικούς τους καταχωρητές και μεταφορά δεδομένων στη CPU.
- Π.χ. Η μονάδα FPU καταλαμβάνει τους opcodes για τις θέσεις συνεπεξεργαστών 1 και 2.
  - Έχει 8 καταχωρητές των 80bit.

Βιβλιογραφία

Χρησιμοποιήθηκε υλικό από παρουσιάσεις των:

- Wayne Wolf, Overheads for Computers as Components 1nd, 2nd, 2008 *(sections 3.1, 3.2, 3.3, 3.4)*

Τέλος Ενότητας