Σετ εντολών του ARM

- Εκδόσεις του ARM.
- Συμβολική γλώσσα (assembly) του ARM.
- Προγραμματιστικό μοντέλο του ARM.
- Οργάνωση μνήμης του ARM.
- Λειτουργίες δεδομένων του ARM.
- Ροή ελέγχου του ARM.

Εκδόσεις του ARM

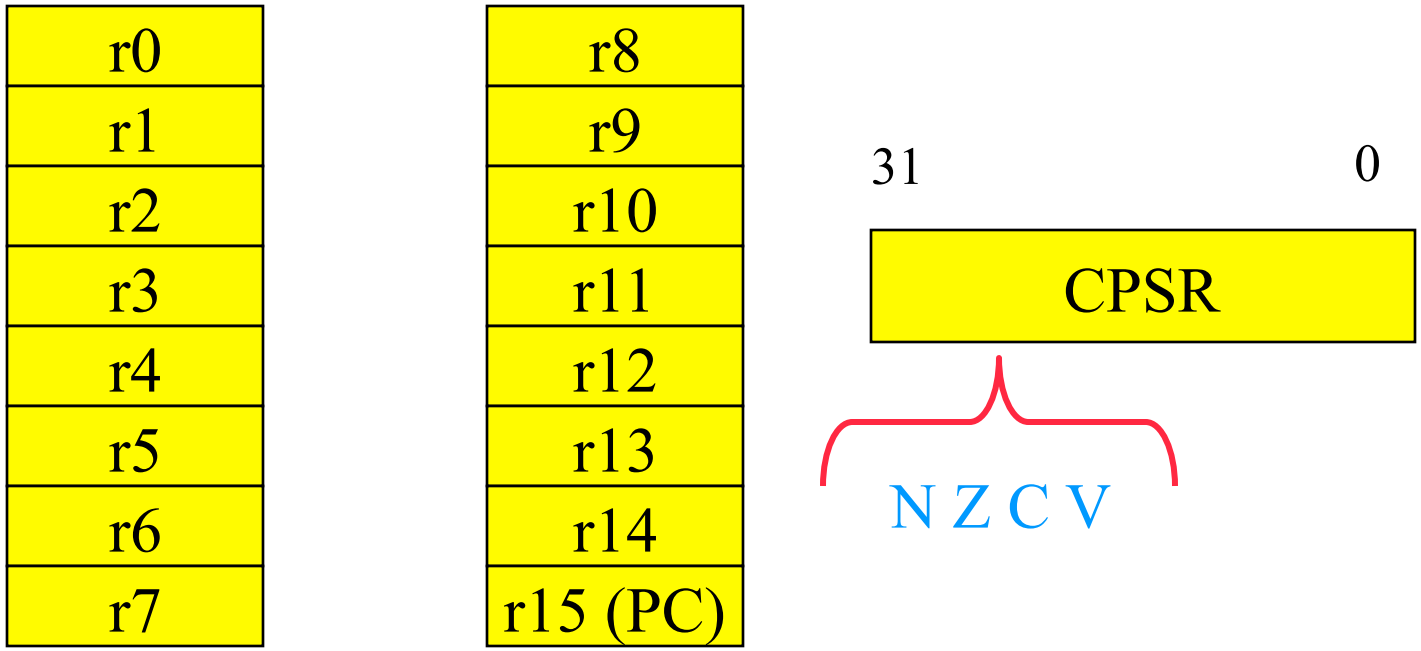
- Η ARM αρχιτεκτονική έχει επεκταθεί σε διάφορες εκδόσεις.
- Θα επικεντρωθούμε στην έκδοση ARM7.

Συμβολική γλώσσα (assembly) του ARM

- Αρκετά τυπική γλώσσα assembly:

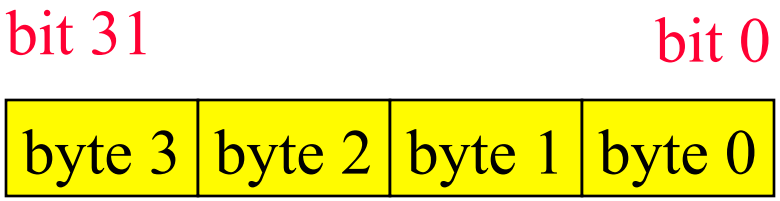
```
        LDR r0,[r8] ; σχόλιο
ετικέτα ADD r4,r0,r1
```

Προγραμματιστικό μοντέλο του ARM

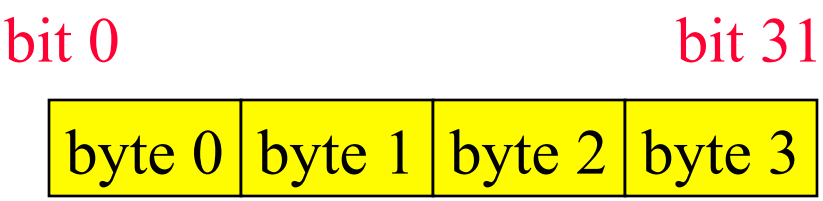


Ακρότατο(Endianness)

- Η σχέση ανάμεσα στη διάταξη των bit και των byte/λέξεων ορίζει το ακρότατο (endianness):



little-endian
μικρό άκρο



big-endian
μεγάλο άκρο

Τύποι δεδομένων του ARM

- Η λέξη έχει μήκος 32.
- Η λέξη μπορεί να διαιρεθεί σε τέσσερα bytes των 8 bits.
- Οι διευθύνσεις του ARM μπορούν να έχουν μήκος 32 bits.
- Η διεύθυνση αναφέρεται στο byte.
  - Η διεύθυνση 4 αρχίζει στο byte 4.
- Στη εκκίνηση μπορεί να οριστεί σαν μικρού ή μεγάλου άκρου (little-endian, big-endian mode).

ARM: bit κατάστασης

- Κάθε πράξη, αριθμητική, λογική ή ολίσθησης θέτει CPSR bits:
  - N (αρνητικό), Z (μηδέν), C (κρατούμενο), V (υπερχείλιση).
- Παραδείγμα:
  - -1 + 1 = 0: NZCV = 0110.
  - 2^{31}-1+1 = -2^{31}: NZCV = 1001.

ARM: εντολές δεδομένων

- Βασική φόρμα:

```
ADD r0,r1,r2
```

  - Υπολογίζει r1+r2, αποθηκεύει στο r0.

- Άμεσοι τελεστές:

```
ADD r0,r1,#2
```

  - Υπολογίζει r1+2, αποθηκεύει στο r0.

ARM: εντολές δεδομένων (2)

- ADD, ADC : πρόσθεση (με ή χωρίς κρατούμενο)
- SUB, SBC : αφαίρεση (με ή χωρίς κρατούμενο)
- RSB, RSC : αντίστροφη αφαίρεση (με ή χωρίς κρατούμενο)
- MUL, MLA : πολλαπλασιασμός (και συσσώρρευση)
- AND, ORR, EOR
- BIC : καθαρισμός bit
- LSL, LSR : λογική μετατόπιση αριστερά/δεξιά
- ASL, ASR : αριθμητική μετατόπιση αριστερά/δεξιά
- ROR : περιστροφή δεξιά
- RRX : περιστροφή δεξιά επεκταμένη με C

Διαφορετικές λειτουργίες δεδομένων

- Λογική μετατόπιση:
  - Συμπληρώνει με μηδενικά.
- Αριθμητική μετατόπιση:
  - Συμπληρώνει με ένα.
- RRX εκτελεί περιστροφή 33-bit, περιλαμβάνοντας C bit του CPSR πάνω από το sign bit.

ARM: εντολές σύγκρισης

- CMP : σύγκριση
- CMN : αρνητική σύγκριση
- TST : bit-wise AND
- TEQ : bit-wise XOR
- Αυτές οι εντολές θέτουν μόνο τα NZCV bits του CPSR.

ARM: εντολές μετακίνησης

- MOV, MVN : μετακίνηση (αρνητική)

```
MOV r0, r1 ; θέτω το r0 στο r1
```

ARM: εντολές φόρτωσης/ αποθήκευσης

- LDR, LDRH, LDRB : φόρτωση (μισή λέξη, byte)
- STR, STRH, STRB : αποθήκευση (μισή λέξη, byte)
- Τύποι διευθυνσιοδότησης:
  - έμμεση με καταχωρητή : `LDR r0,[r1]`
  - με δεύτερο καταχωρητή : `LDR r0,[r1,-r2]`
  - με σταθερά : `LDR r0,[r1,#4]`

ARM: ψευδο-λειτουργία (pseudo-op) ADR

- Δε μπορεί να γίνεται απ’ ευθείας αναφορά σε μία διεύθυνση μέσα σε μια εντολή.
- Δημιουργείται τιμή εφαρμόζοντας αριθμητική στον υπολογιστή.
- Η ψευδο-λειτουργία ADR παράγει εντολή που απαιτείται για να υπολογιστεί η διεύθυνση:

```
ADR r1,FOO
```

Παράδειγμα: Ανάθεση σε C

- C:

```
x = (a + b) - c;
```

- Assembler:

```
ADR r4,a          ; παίρνει διεύθυνση για το a
LDR r0,[r4]       ; παίρνει την τιμή του a
ADR r4,b          ; παίρνει διεύθυνση για το b,
                    επαναχρησιμοποίηση του r4
LDR r1,[r4]       ; παίρνει την τιμή του b
ADD r3,r0,r1      ; υπολογίζει a+b
ADR r4,c          ; παίρνει διεύθυνση για το c
LDR r2,[r4]       ; παίρνει την τιμή του c
```

Ανάθεση σε C, συν.

```
SUB r3,r3,r2     ; πλήρης υπολογισμός του x
ADR r4,x         ; παίρνει διεύθυνση για το x
STR r3,[r4]      ; αποθηκεύει την τιμή του x
```

Παράδειγμα: Ανάθεση σε C

- C:

```
y = a*(b+c);
```

- Assembler:

```
ADR r4,b ; παίρνει διεύθυνση για το b
LDR r0,[r4] ; παίρνει την τιμή του b
ADR r4,c ; παίρνει διεύθυνση για το c
LDR r1,[r4] ; παίρνει την τιμή του c
ADD r2,r0,r1 ; υπολογίζει το μερικό αποτέλεσμα
ADR r4,a ; παίρνει διεύθυνση για το a
LDR r0,[r4] ; παίρνει την τιμή του a
```

Ανάθεση σε C, συν.

```
MUL r2,r2,r0 ; υπολογίζει την τελική τιμή του y
ADR r4,y ; παίρνει διεύθυνση για το y
STR r2,[r4] ; αποθηκεύει το y
```

Παράδειγμα: Ανάθεση σε C

- C:

```
z = (a << 2) |  (b & 15);
```

- Assembler:

```
ADR r4,a ; παίρνει διεύθυνση για το a
LDR r0,[r4] ; παίρνει την τιμή του a
MOV r0,r0,LSL 2 ; εκτελεί μετατόπιση
ADR r4,b ; παίρνει διεύθυνση για το b
LDR r1,[r4] ; παίρνει την τιμή του b
AND r1,r1,#15 ; εκτελεί AND
ORR r1,r0,r1 ; εκτελεί OR
```

Ανάθεση σε C, συν.

```
ADR r4,z ; παίρνει διεύθυνση για το z
STR r1,[r4] ; αποθηκεύει την τιμή για το z
```

Επιπρόσθετοι τύποι διευθυνσιοδότησης

- Διευθυνσιοδότηση βάσης συν σχετική θέση (base-plus-offset):

```
LDR r0,[r1,#16]
```

  - Φορτώνει από την τοποθεσία r1+16
- Η αυτοδεικτοδότηση αυξάνει τον καταχωρητή βάσης:

```
LDR r0,[r1,#16]!
```

- Η μεταδεικτοδότηση εξάγει και μετά προσθέτει τη σχετική θέση:

```
LDR r0,[r1],#16
```

  - Ο r0 φορτώνει από τον r1, μετά προσθέτει 16 στον r1.

ARM: έλεγχος ροής

- Όλες οι λειτουργίες μπορούν να εκτελεστούν υπό συνθήκη, ελέγχοντας τον CPSR:
  - `EQ, NE, CS, CC, MI, PL, VS, VC, HI, LS, GE, LT, GT, LE`
- Λειτουργία διακλάδωσης:

```
B #100
```

  - Μπορεί να εκτελεστεί υπό συνθήκη.

Παράδειγμα: δήλωση if

- C:

```
if (a > b) { x = 5; y = c + d; } else x = c - d;
```

- Assembler:

```
; υπολογίζει και ελέγχει τους όρους
  ADR r4,a ; παίρνει διεύθυνση για το a
  LDR r0,[r4] ; παίρνει την τιμή του a
  ADR r4,b ; παίρνει διεύθυνση για το b
  LDR r1,[r4] ; παίρνει την τιμή του b
  CMP r0,r1 ; συγκρίνει a > b
  BLE fblock ; εάν a <= b, branch to false block
```

δήλωση if, συν.

```
; true block
   MOV r0,#5 ; παράγει τιμή για το x
   ADR r4,x ; παίρνει διεύθυνση για το x
   STR r0,[r4] ; αποθηκεύει το x
   ADR r4,c ; παίρνει διεύθυνση για το c
   LDR r0,[r4] ; παίρνει την τιμή του c
   ADR r4,d ; παίρνει διεύθυνση για το d
   LDR r1,[r4] ; παίρνει την τιμή του d
   ADD r0,r0,r1 ; υπολογίζει το y
   ADR r4,y ; παίρνει διεύθυνση για το y
   STR r0,[r4] ; αποθηκεύει το y
   B after ; branch around false block
```

δήλωση if, συν.

```
; false block
fblock ADR r4,c ; παίρνει διεύθυνση για το c
   LDR r0,[r4] ; παίρνει την τιμή του c
   ADR r4,d ; παίρνει διεύθυνση για το d
   LDR r1,[r4] ; παίρνει την τιμή του d
   SUB r0,r0,r1 ; υπολογίζει c-d
   ADR r4,x ; παίρνει διεύθυνση για το x
   STR r0,[r4] ; αποθηκεύει την τιμή του x
after ...
```

Παράδειγμα: Εφαρμογή εντολής υπό συνθήκη

```
; true block
  MOVLT r0,#5 ; παράγει τιμή για το x
  ADRLT r4,x ; παίρνει διεύθυνση για το x
  STRLT r0,[r4] ; αποθηκεύει το x
  ADRLT r4,c ; παίρνει διεύθυνση για το c
  LDRLT r0,[r4] ; παίρνει την τιμή του c
  ADRLT r4,d ; παίρνει διεύθυνση για το d
  LDRLT r1,[r4] ; παίρνει την τιμή του d
  ADDLT r0,r0,r1 ; υπολογίζει το y
  ADRLT r4,y ; παίρνει διεύθυνση για το y
  STRLT r0,[r4] ; αποθηκεύει το y
```

Εφαρμογή εντολής υπό όρους, συν.

```
; false block
   ADRGE r4,c ; παίρνει διεύθυνση για το c
   LDRGE r0,[r4] ; παίρνει την τιμή του c
   ADRGE r4,d ; παίρνει διεύθυνση για το d
   LDRGE r1,[r4] ; παίρνει την τιμή του d
   SUBGE r0,r0,r1 ; υπολογίζει a-b
   ADRGE r4,x ; παίρνει διεύθυνση για το x
   STRGE r0,[r4] ; αποθηκεύει την τιμή του x
```

Παράδειγμα: δήλωση switch

- C:

```
switch (test) { case 0: … break; case 1: … }
```

- Assembler:

```
   ADR r2,test ; παίρνει διεύθυνση για το test
   LDR r0,[r2] ; φορτώνει την τιμή για το test
   ADR r1,switchtab ; φορτώνει τη διεύθυνση για το
                      switch table
   LDR r1,[r1,r0,LSL #2] ; περιεχόμενο του switch
                           table
switchtab DCD case0
   DCD case1
...
```

Παράδειγμα: φίλτρο FIR

- C:

```
for (i=0, f=0; i<N; i++)
  f = f + c[i]*x[i];
```

- Assembler

```
; κώδικας έναρξης βρόχων
  MOV r0,#0 ; χρησιμοποιεί το r0 για το i
  MOV r8,#0 ; χρησιμοποιεί το r8 για τις διαδοχικές
            ; θέσεις μνήμης
  ADR r2,N ; παίρνει διεύθυνση για το N
  LDR r1,[r2] ; παίρνει την τιμή του N
  MOV r2,#0 ; χρησιμοποιεί το r2 για το f
```

φίλτρο FIR, συν.

```
   ADR r3,c ; φορτώνει το r3 με τη βάση του c
   ADR r5,x ; φορτώνει το r5 με τη βάση του x
; loop body
loop LDR r4,[r3,r8] ; παίρνει το c[i]
   LDR r6,[r5,r8] ; παίρνει το x[i]
   MUL r4,r4,r6 ; υπολογίζει c[i]*x[i]
   ADD r2,r2,r4 ; προσθέτει στο τρέχον αποτέλεσμα
   ADD r8,r8,#4 ; προσθέτει μία λέξη που
                  αντισταθμίζεται στο περιεχόμενο της
                  σειράς
   ADD r0,r0,#1 ; προσθέτει το 1 στο i
   CMP r0,r1 ; έξοδος?
   BLT loop ; εάν i < N, συνεχίζει
```

ARM: σύνδεση υπορουτινών

- Εντολή διακλάδωσης και σύνδεσης:

```
BL foo
```

  - Αντιγράφει τον παρόντα PC στο r14.
- Για να επιστρέψουμε από την υπορουτίνα:

```
MOV r15,r14
```

Κλήση εμφωλευμένων υπορουτινών

- Η εμφώλευση/επαναληπτικότητα απαιτεί συμβατική κωδικοποίηση:

```
f1      LDR r0,[r13] ; φορτώνει arg στο r0 από τη
                       στοίβα
        ; κλήση της f2()
        STR r14,[r13]! ; φορτώνει τη διεύθυνση
                         επιστροφής του f1
        STR r0,[r13]! ; φορτώνει arg στο f2 στη στοίβα
        BL f2 ; διακλάδωση και σύνδεση στο f2
        ; επιστρέφει από το f1()
        SUB r13,#4 ; τραβάει το f2's arg από τη στοίβα
        LDR r13!,r15 ; αποκαθιστά τον καταχωρητή και
                       επιστρέφει
```

Σύνοψη

- Αρχιτεκτονική φόρτωσης αποθήκευσης
- Οι περισσότερες εντολές είναι τύπου RISC, και εκτελούνται σε έναν κύκλο.
  - Κάποιες εντολές που χρησιμοποιούν πολλούς καταχωρητές παίρνουν περισσότερο χρόνο.
- Όλες οι εντολές μπορούν να εκτελεστούν υπό συνθήκη.