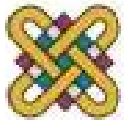


Πανεπιστήμιο Δυτικής Μακεδονίας
Τμήμα Μηχανικών Πληροφορικής & Τηλεπικοινωνιών

Ενσωματωμένα Συστήματα

Ενότητα 4: Επεξεργαστές & Σύνολα Εντολών. Κεντρικές Μονάδες Επεξεργασίας. Εισαγωγή στην αρχιτεκτονική ARM.

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 παρουσίαση της δημοφιλούς αρχιτεκτονικής ARM.
- Η εισαγωγική περιγραφή της ARM ISA.

Σύνολα εντολών (ISA)

- Χρησιμοποιούνται για την ταξινόμηση στην Αρχιτεκτονική Υπολογιστών.
- Σημαντικό στοιχείο: Γλώσσα Assembly.
- ISA: Διασύνδεση (interface) του προγραμματιστή με το υλικό.
- Αν και ο προγραμματισμός γίνεται σε υψηλού επιπέδου γλώσσες, το σύνολο εντολών είναι το κλειδί στην ανάλυση της απόδοσης.

Αρχιτεκτονική von Neumann (1)

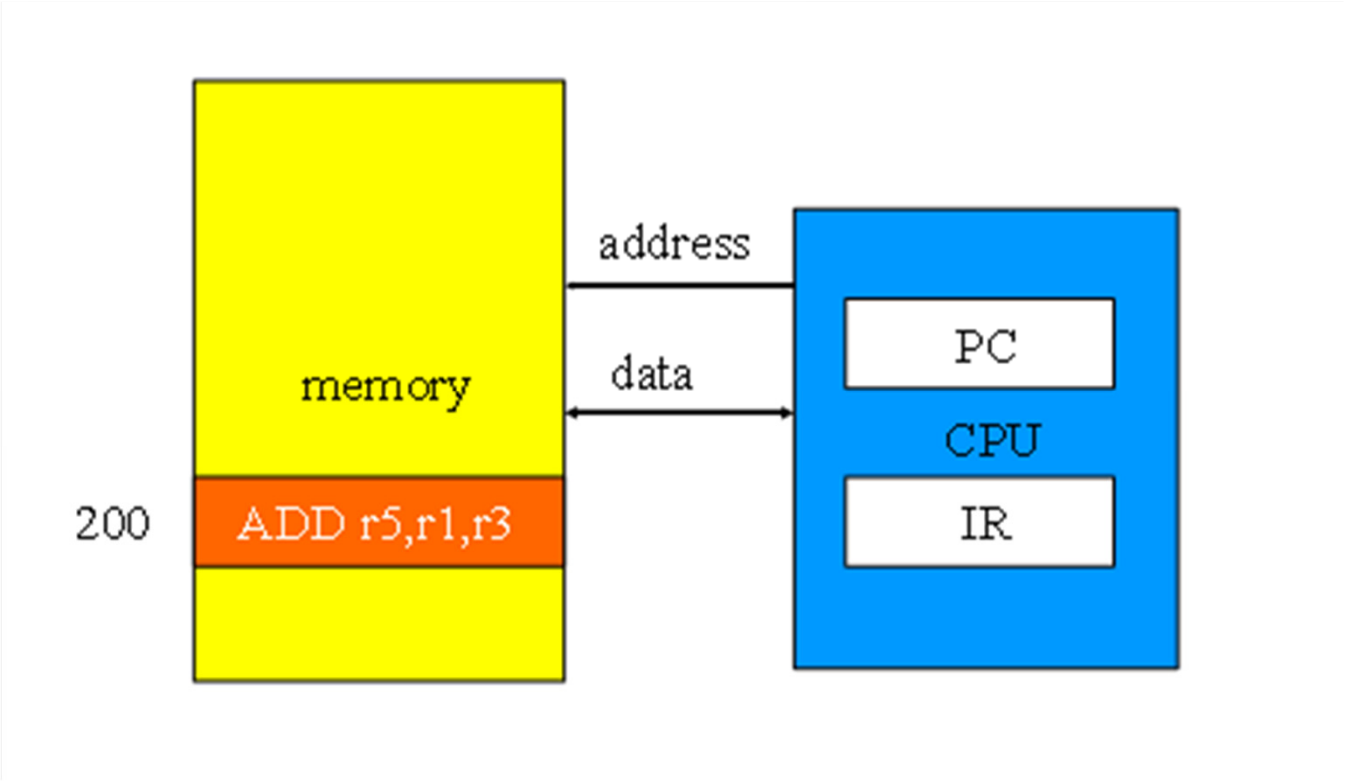
- Η μνήμη διατηρεί τα δεδομένα & τις οδηγίες.
- Κεντρική μονάδα επεξεργασίας (CPU) ανακαλεί εντολές από τη μνήμη.
  - Ξεχωριστή CPU και μνήμη: ξεχωρίζει έναν υπολογιστή που μπορούμε να προγραμματίσουμε.
- Οι CPU καταχωρητές υποστηρίζουν:
  - μετρητή προγράμματος (PC),
  - καταχωρητή εντολών (IR),
  - γενικού σκοπού καταχωρητές, κ.α..

  Von Neuman: Η ίδια μνήμη φυλάσσει και δεδομένα και εντολές.

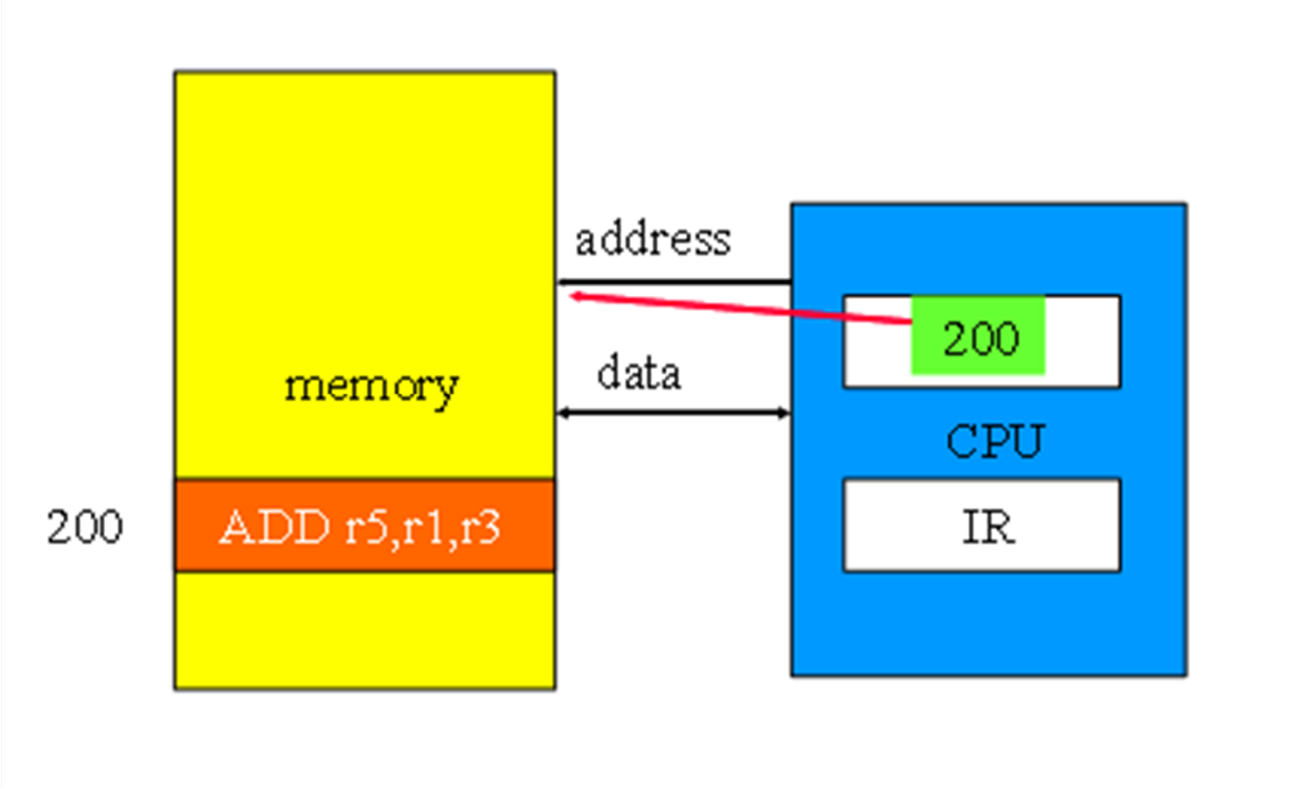
Αρχιτεκτονική von Neumann (2)

- Βασικό στοιχείο της αρχιτεκτονικής von neumann είναι ένας καταχωρητής, που ονομάζεται **Μετρητής Προγράμματος** (Program Counter).
- Ο μετρητής προγράμματος δεν καθορίζει άμεσα τι κάνει η μηχανή, αλλά έμμεσα με την κατάδειξη μιας εντολής στη μνήμη.
- Αυτός είναι ο διαχωρισμός ενός υπολογιστή αποθηκευμένου προγράμματος (stored program computer) από μια γενική μηχανή πεπερασμένων καταστάσεων (finite state machine).

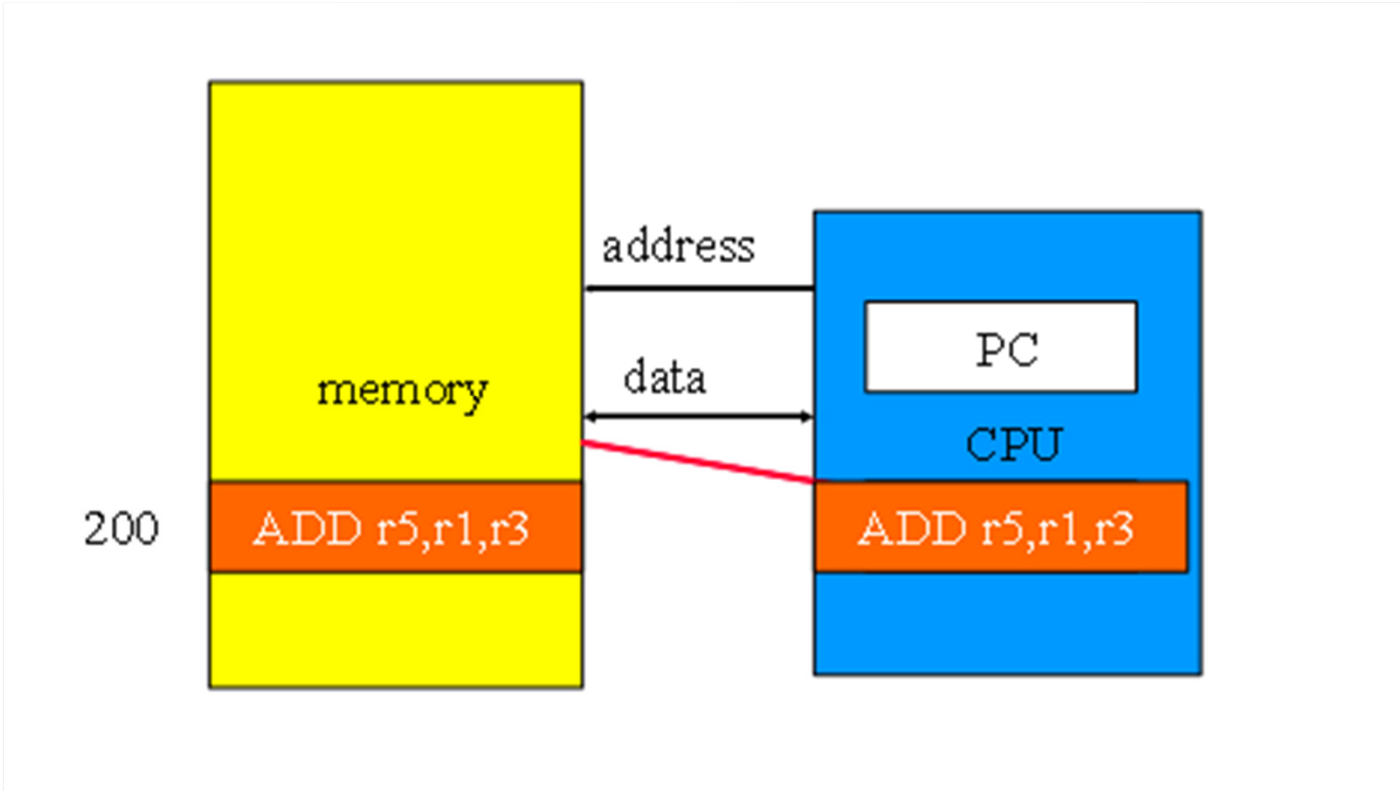
CPU + memory (1)



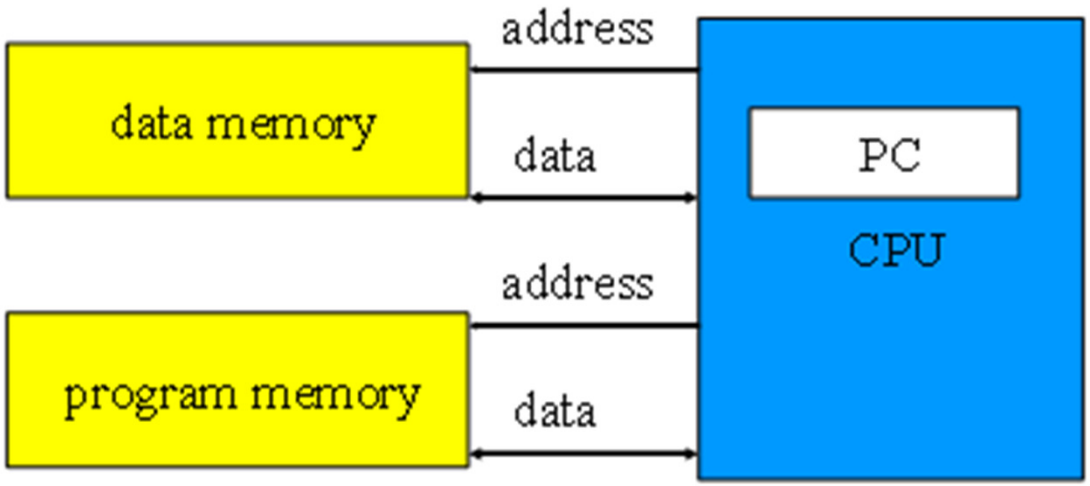
CPU + memory (2)



CPU + memory (3)



Αρχιτεκτονική Harvard



- **Εναλλακτικό στυλ του Von Neuman.**
- **Δυο ξεχωριστές μνήμες, μια για τις εντολές και μια για τα δεδομένα.**
- **Είναι δύσκολη η εγγραφή αυτοτροποποιούμενων προγραμμάτων.**

von Neumann vs. Harvard

- Η Harvard δεν μπορεί να χρησιμοποιήσει αυτό-τροποποιούμενο κώδικα.
- Η Harvard επιτρέπει δύο ταυτόχρονες προσκομίσεις από τη μνήμη.
- Οι περισσότερες DSPs χρησιμοποιούν Harvard αρχιτεκτονική για συνεχή ροή δεδομένων :
  - μεγαλύτερο εύρος ζώνης της μνήμης.
  - πιο προβλέψιμο εύρος ζώνης (μετακίνηση δεδομένων στους κατάλληλους “συγχρονισμένους” χρόνους).

RISC vs. CISC

- Πολύπλοκο σετ εντολών υπολογιστή (**CISC**):
  - Πρώιμες αρχιτεκτονικές.
  - Πολλοί τρόποι διευθυνσιοδότησης.
  - Πολλές λειτουργίες (π.χ. Αναζήτηση συμβολοσειράς).
- Μειωμένο σετ εντολών υπολογιστή (**RISC**):
  - Εξέλιξη στην αρχιτεκτονική υπολογιστών.
  - Φόρτωση /αποθήκευση.
  - Διασωληνωμένες οδηγίες.
  - Αρχικά οι RISC είχαν πολύ μεγαλύτερη ταχύτητα από CISC.
  - Χρησιμοποιήθηκαν τεχνικές RISC σε CISC, οπότε το χάσμα απόδοσης έχει μικρύνει.

Χαρακτηριστικά των εντολών

- Σταθερού ή μεταβλητού μήκους.
- Τρόποι Διευθυνσιοδότησης.
- Αριθμός τελεστών.
- Τύποι τελεστών.

Μοντέλο προγραμματισμού

- Μοντέλο προγραμματισμού: καταχωρητές ορατοί στον προγραμματιστή.
- Ορισμένοι καταχωρητές δεν είναι ορατοί (π.χ. IR).
- **Ένας επεξεργαστής έχει πληθώρα καταχωρητών. Κάποιοι καταχωρητές είναι ορατοί στο χρήστη και κάποιοι είναι αόρατοι.**

Πολλαπλές υλοποιήσεις

- Επιτυχείς αρχιτεκτονικές έχουν αρκετές υλοποιήσεις, ως προς:
- ποικίλες ταχύτητες ρολογιού,
- διαφορετικό πλάτος διαύλου,
- διαφορετικά μεγέθη μνήμης cache,
- κ.α.

Η ίδια αρχιτεκτονική επεξεργαστή μπορεί να υλοποιηθεί με αρκετές διαφοροποιήσεις!

Γλώσσα Assembly

- Συμβολικά ονόματα assembly-προς-μια-εντολή που εκτελείται.
- Βασικά χαρακτηριστικά:
  - Μία εντολή ανά γραμμή.
  - Οι ετικέτες παρέχουν τα ονόματα για τις διευθύνσεις (συνήθως στην πρώτη στήλη).
  - Οι οδηγίες συχνά ξεκινούν στις μετέπειτα στήλες.

Παράδειγμα ARM assembly

```
Label1      ADR r4,c
            LDR r0,[r4] ; a comment
            ADR r4,d
            LDR r1,[r4]
            SUB r0,r0,r1 ; comment
```

Ο τρόπος που γράφεται η συμβολική γλώσσα οφείλεται στους πρώτους συμβολομετραφραστές που και αυτοί είχαν γραφτεί στη συμβολική γλώσσα και έπρεπε να χωρούν σε μια πολύ μικρή ποσότητα μνήμης.

Ψευδό-λειτουργίες

- Ορισμένες οδηγίες assembler δεν αντιστοιχούν άμεσα σε οδηγίες. Παραδείγματα:
  - Ορισμός τρέχουσας διεύθυνσης.
  - Κράτηση θέσης μνήμης για μεταβλητές.
  - Σταθερές.

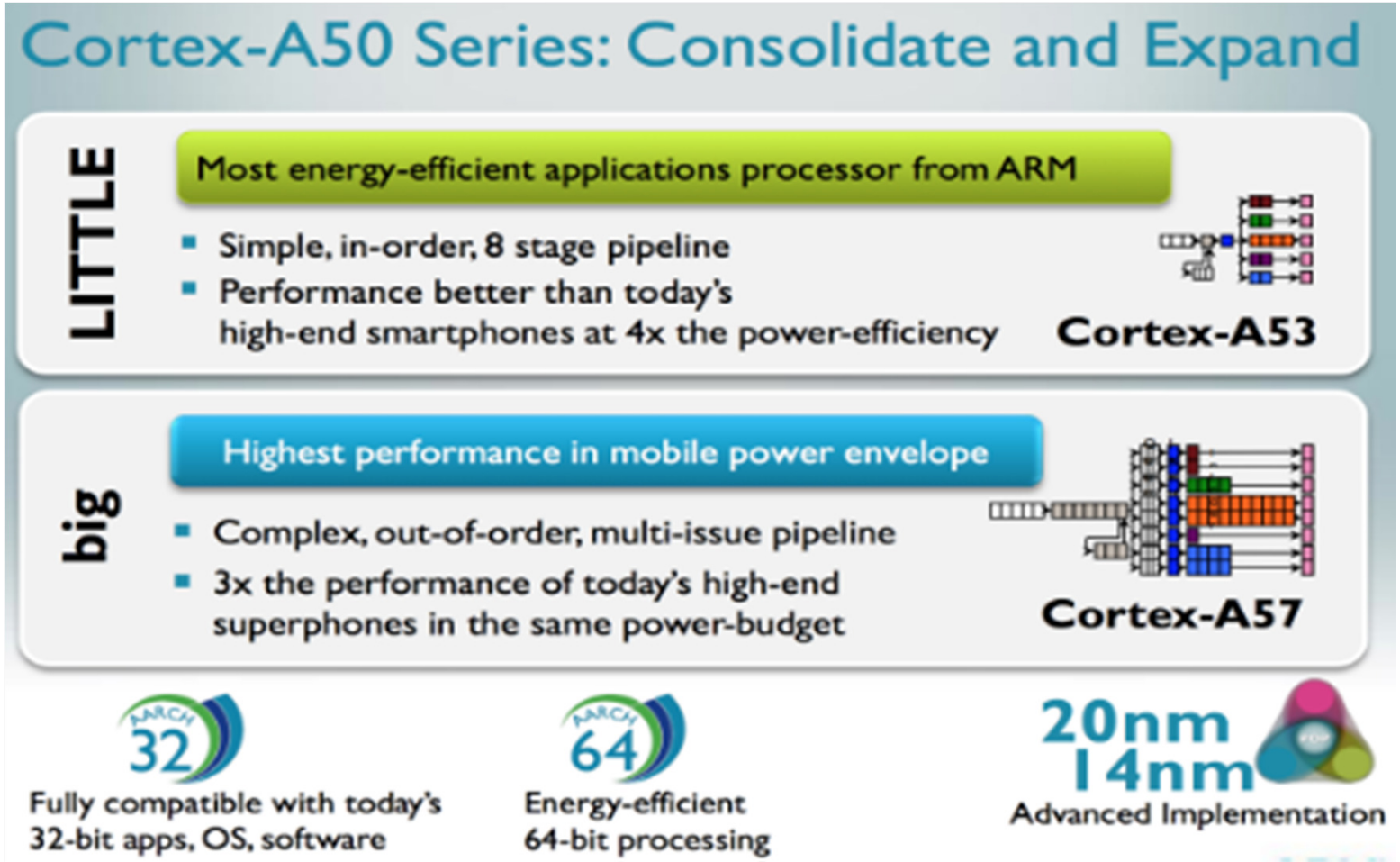
Ψευδολειτουργίες που βοηθούν τους προγραμματιστές να δημιουργούν ολοκληρωμένα προγράμματα συμβολικής γλώσσας.

Επεξεργαστές ARM

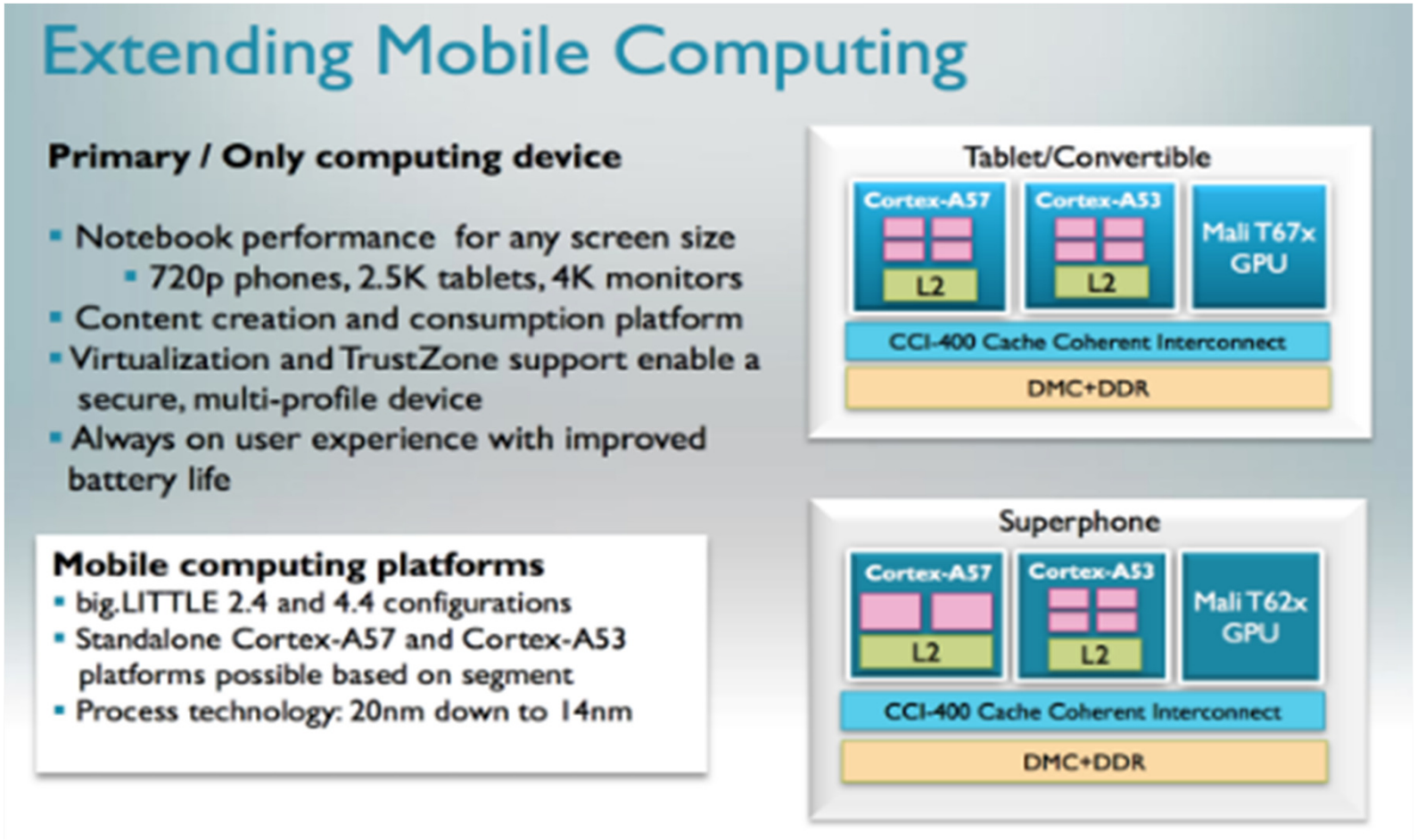
NVIDIA's Tegra in the flesh, booting to Android and pumping out 1080p video

Model number	Semiconductor technology	CPU instruction set	CPU	GPU	Memory technology	Availability	Utilizing Devices
Tegra 250 AP20H	40 nm	ARMv7	1 GHz dual-core ARM Cortex-A9	ULP GeForce 300 MHz	Single-channel LPDDR2 600 MHz or DDR2 667 MHz	Q1 2010	LG Optimus 2X, Motorola Atrix 4G, Motorola Droid X2, Motorola Photon, Samsung Galaxy R, Samsung Captivate Glide, Tesla Model S, ZTE Mimosa X, Micromax Superfone A85
Tegra 250 T20	40 nm	ARMv7	1 GHz dual-core ARM Cortex-A9	ULP GeForce 333 MHz	Single-channel LPDDR2 600 MHz or DDR2 667 MHz	Q1 2010	Acer Iconia Tab A100, A200 and A500, Asus Slider, LG Optimus Pad, Atomic Design Tomorrow Processor Board,[17] Eseer EasyPad, Notion Ink Adam tablet, Olivetti OliPad 100, Point of View Mobii 10.1, ViewSonic G Tablet, Motorola Xoom,[17] Toshiba AC100, Toshiba Folio 100, ASUS Eee Pad Transformer, Advent Vega, Hannspree Hannspad, Aigo N700, CompuLab Trim-Slice nettop, Dell Streak 7, E-Noa Interpad, Malata Tablet Zpad, MSI 10-inch (250 mm) tablet, Toradex Colibri Tegra 2, Toshiba Thrive tablet, Samsung Galaxy Tab 10.1, T-Mobile G Slate, Lenovo IdeaPad Tablet K1, Lenovo ThinkPad Tablet, Velocity Micro Cruz Tablet L510, Dell Streak Pro,[18] Zyrex Onepad SP1110G, Zyrex Onepad SP1115G, Sony Tablet S

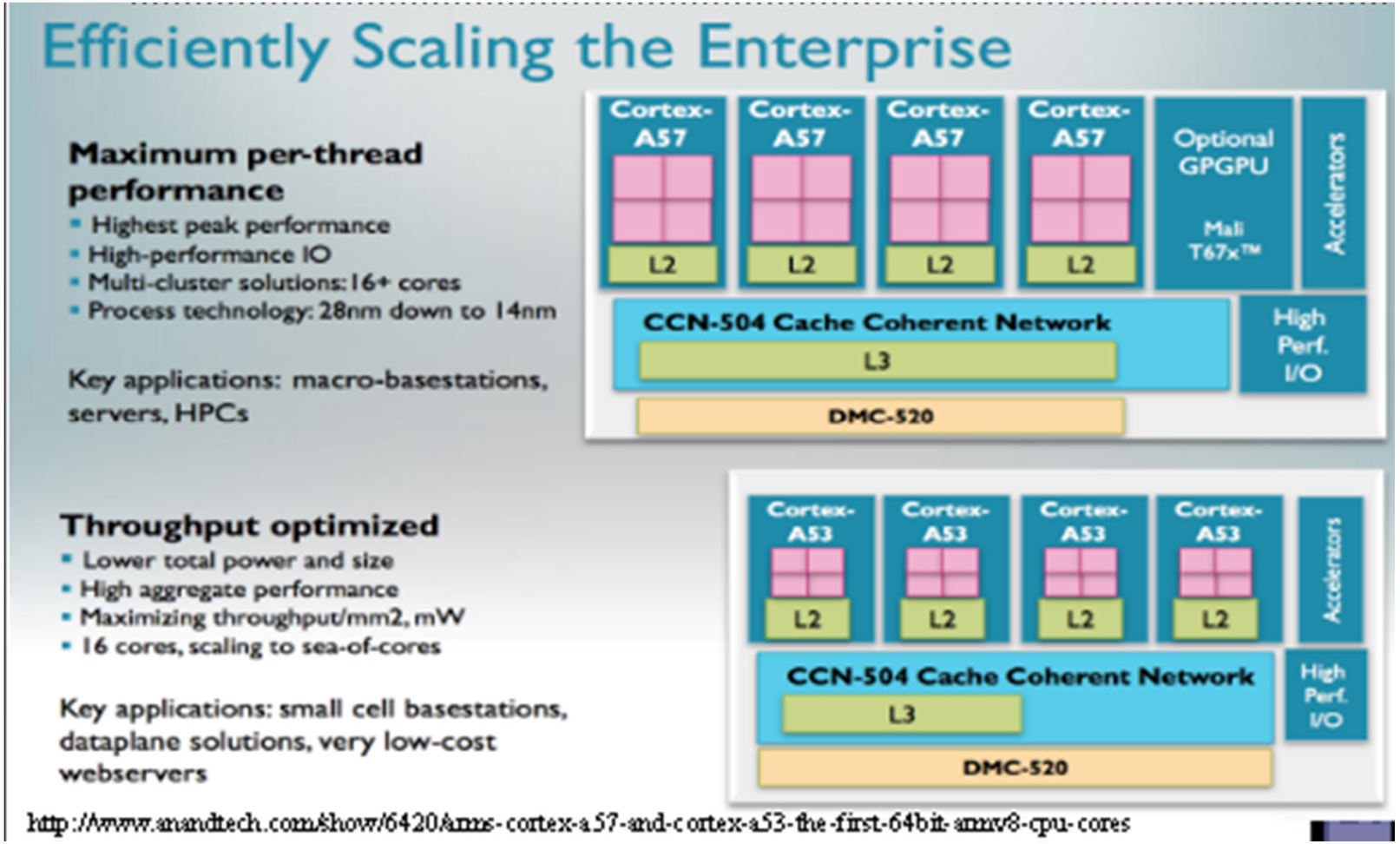
Ο πρώτος 64bit επεξεργαστής ARM -ATLAS (@2014)



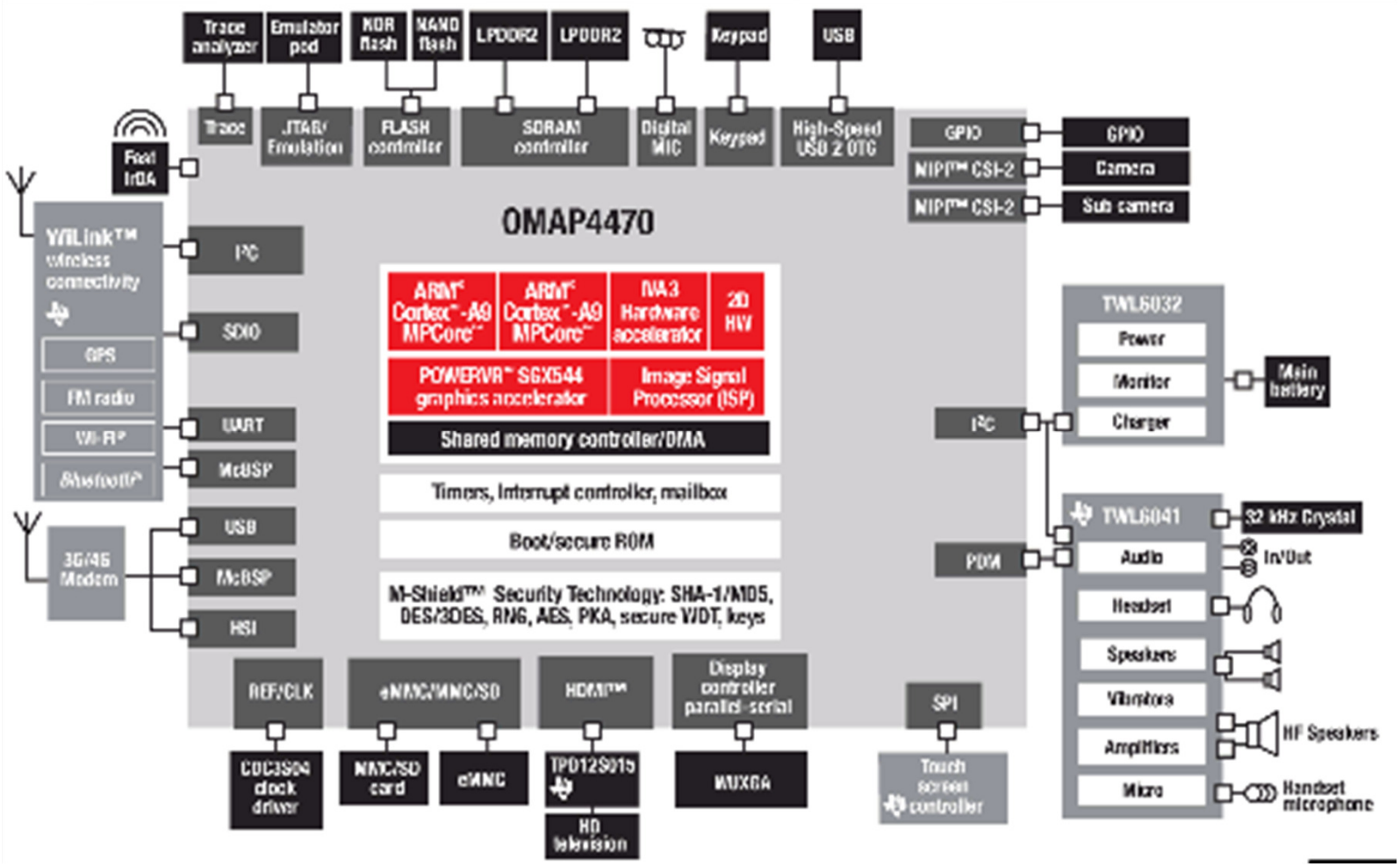
Η χρήση του ARM στις φορητές συσκευές



Η αποδοτικότητα του ARM 64bit



Η δημοφιλής αρχιτεκτονική Texas Instruments OMAP 4 (1/3)



Η δημοφιλής αρχιτεκτονική Texas Instruments OMAP 4 (2/3)

- Σχεδιασμένο για να χρησιμοποιηθεί σε smartphones, ταμπλέτες και άλλες κινητές συσκευές πλούσιες σε πολυμέσα.
- Οι επιταχυντές υλικού IVA 3 επιτρέπουν την πλήρη προβολή HD 1080p, πολυ-πρότυπων βίντεο (κωδικοποίηση / αποκωδικοποίηση).
- Ταχύτερη, υψηλότερη ποιότητα εικόνας και καταγραφής βίντεο με την ψηφιακή φωτογραφική μηχανή SLR απεικόνισης έως και 20 megapixels.

Η δημοφιλής αρχιτεκτονική Texas Instruments OMAP 4 (3/3)

- Dual-core ARM® Cortex™-A9 MPCore™ με συμμετρική πολυεπεξεργασία (SMP).
- Ο ενσωματωμένος PowerVR SGX540 ™ επιταχυντής γραφικών πραγματοποιεί 3D gaming και 3D user διασυνδέσεις.
- Ιδιαίτερα βελτιστοποιημένη κινητή πλατφόρμα εφαρμογών.
- Η OMAP4430 λειτουργεί μέχρι και 1 Ghz.
- Η OMAP4460 λειτουργεί μέχρι και 1.5 Ghz.

Γενικά Στοιχεία

- Ο ARM είναι RISC επεξεργαστής.
- Χρησιμοποιείται για μικρές σε μέγεθος και υψηλής σε απόδοση, εφαρμογές.
- Η αρχιτεκτονική του είναι απλή και οδηγεί σε μικρές υλοποιήσεις με χαμηλή κατανάλωση ισχύος.

Ιστορικά Στοιχεία (1/2)

- 1985: Η εταιρία Acorn Computer Group κατασκευάζει τον πρώτο RISC επεξεργαστή για εμπορική χρήση. (Comparable with 80286, only 25000 transistors)
- 1990: Από τη συνεργασία των Acorn και Apple δημιουργείται η Advanced RISC Machines (A.R.M.).
- 1991: Ο πρώτος embeddable RISC επεξεργαστής, ο ARM6.
- 1992 – 1994: Διάφορες εταιρίες ζητούν άδεια για να χρησιμοποιήσουν τον ARM (Sharp, Shamsung), ενώ το 1993 παράγεται ο ARM7, ο πρώτος επεξεργαστής για multimedia εφαρμογές.

Ιστορικά Στοιχεία (2/2)

- 1995: Δημιουργείται η Αρχιτεκτονική Thumb και η οικογένεια ARM8.
- 1996 – 2000: Η Alcatel, η Huyndai, η Philips, η Sony, ζητούν άδεια για χρήση του ARM, ενώ το 1999 η ARM συνεργάζεται με την Erickson για την κατασκευή του Bluetooth.
- 2000 – 2002: Το μερίδιο στην αγορά της ARM στους 32 – bit embedded RISC επεξεργαστές φτάνει το 80%. Δημιουργείται το πρόγραμμα ARM Developer Suite.

Ιστορία της αρχιτεκτονικής ARM (1)

Version	Year	Features	Implementations
v1	1985	The first commercial RISC (26-bit)	ARM1
v2	1987	Coprocessor support	ARM2, ARM3
v3	1992	32-bit, MMU, 64-bit MAC	ARM6, ARM7
v4	1996	Thumb	ARM7TDMI, ARM8, ARM9TDMI, StrongARM
v5	1999	DSP and Jazelle extensions	ARM10, XScale
v6	2001	SIMD, Thumb-2, TrustZone, multiprocessing	ARM11, ARM11 MPCore
v7	?	VFP-3	?

Ιστορία της αρχιτεκτονικής ARM (2)

Architecture	Family
ARMv1	ARM1
ARMv2	ARM2, ARM3
ARMv3	ARM6, ARM7
ARMv4	StrongARM, ARM7TDMI, ARM9TDMI
ARMv5	ARM7EJ, ARM9E, ARM10E, XScale
ARMv6	ARM11, ARM Cortex-M
ARMv7	ARM Cortex-A, ARM Cortex-M, ARM Cortex-R
ARMv8	*No cores available yet. Will support 64-bit data and addressing* [18][19]

- **"Application" profile: Cortex-A series.**
- **"Real-time" profile: Cortex-R series.**
- **"Microcontroller" profile: Cortex-M series.**

Η εταιρία ARM

ARM (UK Company):

- Επιχειρηματικό Μοντέλο: Έκδοση αδειών για την χρήση της αρχιτεκτονικής ARM IP.
  - Άδεια υλοποίησης:
    - Soft core (μπορεί να χρησιμοποιηθεί για οποιαδήποτε διαδικασία, αλλά όχι βελτιστοποιημένη).
    - Hard core (βελτιστοποιημένη για ειδική παραγωγική διαδικασία).
  - Άδεια Αρχιτεκτονικής:
    - Άλλες εταιρίες παράγουν πυρήνες συμβατούς με ARM ISA.

95% των ενσωματωμένων επεξεργαστών που πωλούνται είναι ARM!

Windows RT (~windows 8) supports ARM

Η ARM προσπάθησε να σχεδιάσει κάποτε το δικό της PC..

- Acorn Archimedes.
- Περιορισμένη επιτυχία.
- Ακόμα κι αν ήταν ταχύτερο από τη Motorola 68000, απέτυχε εξαιτίας του IBM PC.

A3010

Type	Personal computer or Home computer
Release date	June 1987
Discontinued	mid-1990s
Operating system	RISC OS or RISC iX
CPU	ARM
Memory	512 KB–16 MB

Η ARM κυριαρχεί στην αγορά ολοκληρωμένων συστημάτων

- Από το 2009, οι επεξεργαστές ARM αντιπροσωπεύουν περίπου το 90% του συνόλου ενσωματωμένων 32-bit επεξεργαστών RISC και χρησιμοποιούνται ευρέως στα καταναλωτικά ηλεκτρονικά προϊόντα, συμπεριλαμβανομένων των :
  - personal digital assistants (PDAs),
  - tablets,
  - κινητά τηλέφωνα,
  - ψηφιακών πολυμέσων και συσκευές αναπαραγωγής μουσικής,
  - φορητές κονσόλες παιχνιδιών,
  - αριθμομηχανές και
  - περιφερειακά υπολογιστών, όπως σκληρούς δίσκους και δρομολογητές.

Γενικά για τον ARM (1/2)

ΣΤΟΧΟΣ: το απλό design

- Load – store architecture
- 32 bit data bus
- 3 τρόποι διευθυνσιοδότησης

Ένα τυπικό τσιπ ARM αποτελείται από:

- Ελεγκτές Περιφερειακών.
- Ψηφιακό επεξεργαστή σήματος.
- Ενσωματωμένο chip μνήμης.
- Επεξεργαστή ARM.

Γενικά (2/2)

Απλή αρχιτεκτονική
+
Απλό instruction set
+
Πυκνότητα κώδικα

Μικρό μέγεθος

Μικρή κατανάλωση ισχύος

Πλεονεκτήματα ARM

- Μικρό μέγεθος (# τρανζίστορ).
- Ελάχιστη κατανάλωση ενέργειας.
- Εξαιρετικά αρθρωτή αρχιτεκτονική (εύκολο να οικοδομηθούν υλοποιήσεις που βασίζονται σε συγκεκριμένους ARM επεξεργαστές, το μόνο υποχρεωτικό στοιχείο είναι σταθερή διασωλήνωση MMU, cache, FPU είναι προαιρετικά).
- High Performance (e.g. PXA255 Xscale @ 400Mhz ίδια απόδοση με Pentium 2@300Mhz, με το 1/50 της κατανάλωσης ενέργειας).

Εκδόσεις ARM

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

- **Οικογένεια επεξεργαστών RISC.**
- **Η ARM δεν κατασκευάζει τα τσιπ (fabless). Χορηγεί άδειες για την αρχιτεκτονική σε άλλες εταιρίες.**
- **32bit word length.**
- **ARM7 Von Neumann.**
- **ARM9 Harvard.**
- **Η διαφορά είναι:**

ARM είναι μια RISC CPU

- RISC CPU.
- Load-store αρχιτεκτονική (λειτουργίες μόνο από και προς τους καταχωρητές).
- Δεν υπάρχει υποστήριξη για μη ευθυγραμμισμένες προσπελάσεις μνήμης.
- Ενιαίο 16 X 32-bit αρχείο καταχωρητών.
- Οδηγίες: 32bit (οι περισσότερες με 3 τελεστές κωδικοποίησης), για να διευκολύνει την αποκωδικοποίηση και τη διασωλήνωση, με το κόστος της μειωμένης πυκνότητας κώδικα.
- Κυρίως εκτέλεση μονού κύκλου.
- Πολύ καλή διασωλήνωση.

ARM is a RISC CPU

- Απλός σχεδιασμός:
  - Δεν χρησιμοποιεί τους καταχωρητές παραθύρου / μετονομασίας όπως οι άλλοι RISC.
  - Αυτόματη δεικτοδότηση (αυτόματη αύξηση/μείωση σε 2 κύκλους πρόσβασης μνήμης).
  - Πολλαπλές οδηγίες μεταφοράς καταχωρητών (φόρτωση / αποθήκευση έως και 16 καταχωρητές ταυτόχρονα --- όχι 1 κύκλο/εντολή).
- Υπό όρους εκτέλεση των περισσότερων εντολών, μείωση της επιβάρυνσης διακλάδωσης και την αντιστάθμιση για την έλλειψη πρόβλεψης διακλαδώσεων.
- 32-bit ολισθητής βαρελιού (ολισθαίνει μια λέξη δεδομένων κατά ένα συγκεκριμένο αριθμό bits σε ένα κύκλο ρολογιού).

Γλώσσα assembly στους ARM

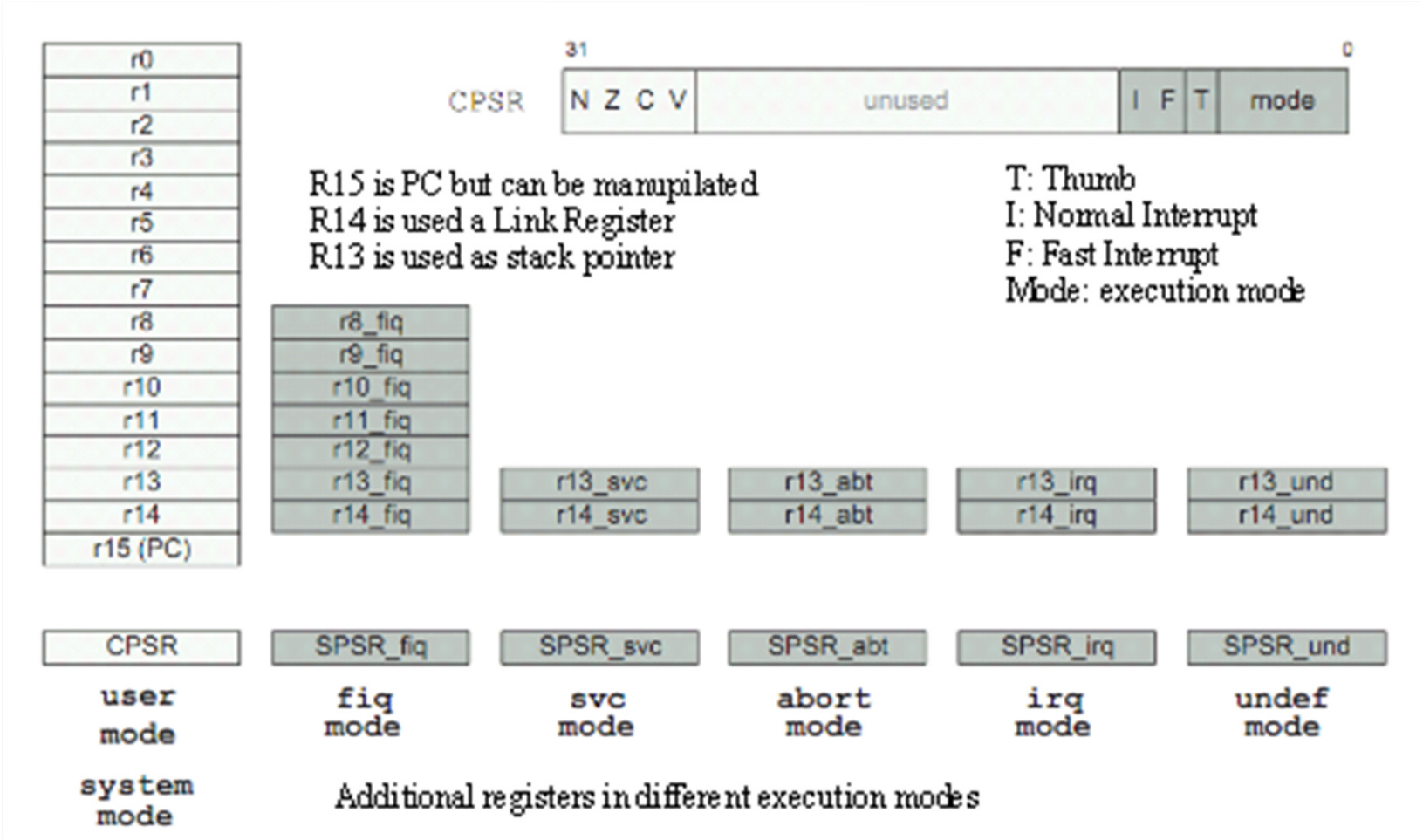
Αρκετά τυπική συμβολική γλώσσα:

```
    LDR r0,[r8] ; a comment
Label ADD r4,r0,r1
```

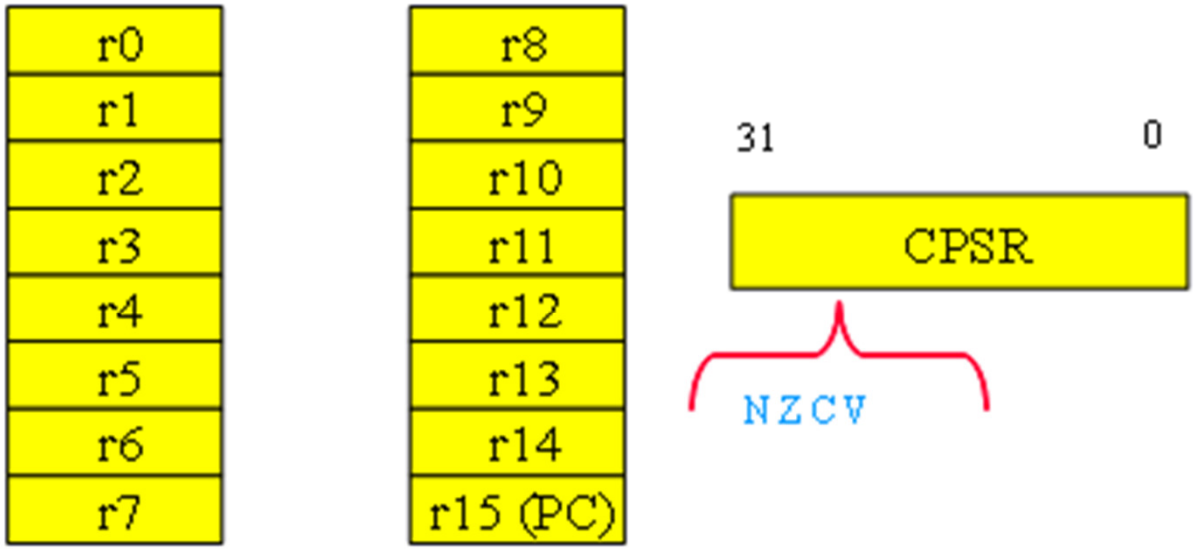
Καταχωρητές

- 32 καταχωρητές γενικής χρήσης.
- Ο ARM μπορεί να λειτουργήσει σε διάφορές καταστάσεις λειτουργίας.
- Σε κάθε λειτουργία είναι ορατοί διαφορετικοί καταχωρητές.

Οι ορατοί καταχωρητές στον ARM

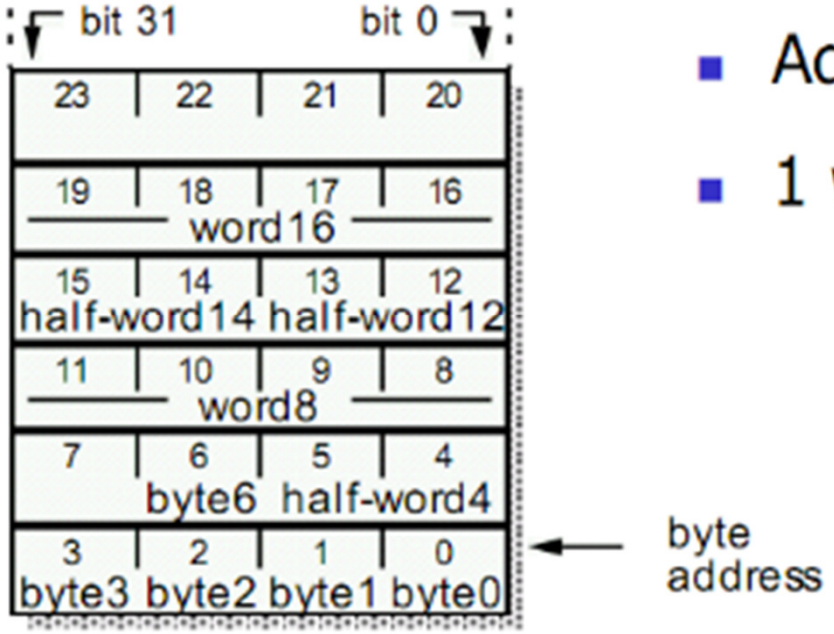


Μοντέλο προγραμματισμού σε ARM



CPSR: Current Program Status Register

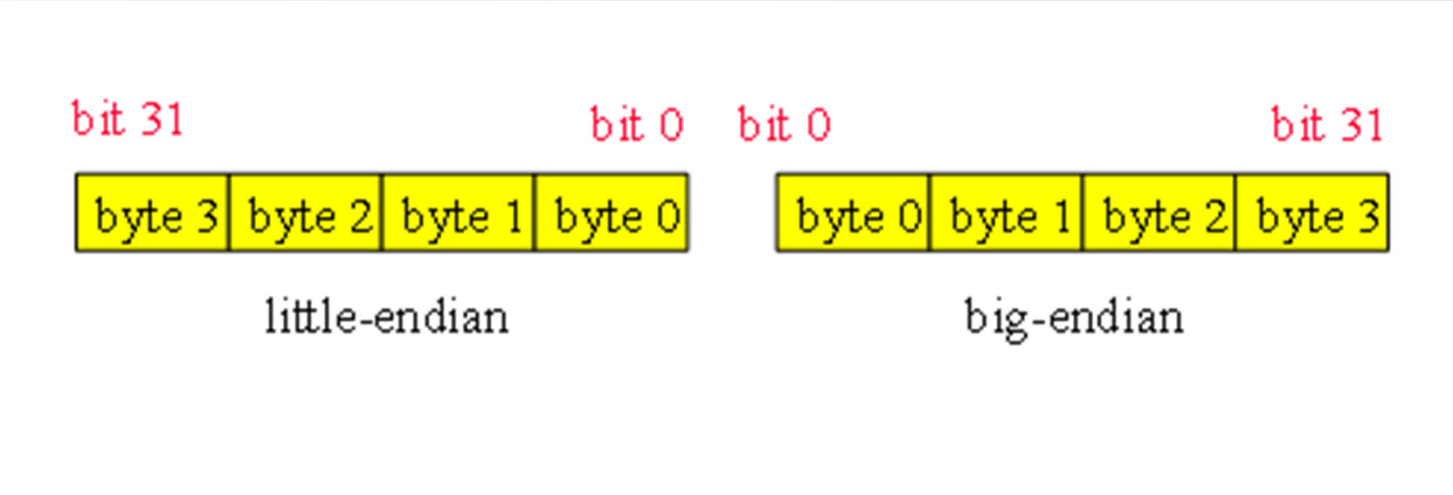
Οργάνωση μνήμης



- Address bus: 32 – bits
- 1 word = 32 – bits

Endianness

- Η σχέση μεταξύ bit και byte/word ordering προσδιορίζει το endianness:



Οι τύποι δεδομένων του ARM

- Η λέξη έχει μήκος 32 bit.
- Η λέξη μπορεί να διαιρεθεί σε τέσσερα 8-bit (bytes).
- Οι ARM διευθύνσεις έχουν 32bits μέγεθος.
- Η διεύθυνση αναφέρεται σε byte.
- π.χ. η διεύθυνση 4 αρχίζει στο byte 4.
- Μπορεί να διαμορφωθεί στην έναρξη, είτε ως big- είτε ως little-endian.

Τα bit κατάστασης του ARM

- Κάθε αριθμητική, λογική, ή αλλαγή λειτουργίας καθορίζει τα CPSR bits:
  - N (αρνητικό), Z (μηδέν), C (carry), V (overflow).
- Παράδειγμα:
  - -1 + 1 = 0: NZCV = 0110.
  - 2^{31}-1+1 = -2^{31}: NZCV = 0101.

Οι εντολές δεδομένων του ARM

- Βασική μορφή:
  - **ADD r0, r1, r2**
    - Computes r1+r2, stores in r0.
- Άμεσος τελεστής(άμεσοι τελεστέοι):
  - **ADD r0, r1, #2**
    - r1+2 και αποθήκευση στο r0.

Εντολές

- Τρεις κατηγορίες εντολών
  - Επεξεργασία δεδομένων.
  - Μεταφορά δεδομένων.
  - Ροή ελέγχου.

Λειτουργία επιτήρησης

- Στο user mode τις διαδικασίες έξω από αυτό τις αναλαμβάνει αποκλειστικά το λειτουργικό σύστημα.
- Με “supervisor calls”, ο χρήστης περνά στο system level έχοντας τη δυνατότητα να επηρεάζει λειτουργίες του συστήματος.

I/O System

- Ο ARM χειρίζεται τα περιφερειακά ως “συσκευές αντιστοιχισμένες στη μνήμη με support καταχωρητές”.

- Καταχωρητές:
  - IRQ : Απλοί καταχωρητές.
  - FIQ : Γρήγοροι καταχωρητές.

Εξαιρέσεις

- Εξαιρέσεις:
  - Καταχωρητές.
  - Κλήση supervisor.
  - Παγίδες.
- Όταν συμβαίνει εξαίρεση:
  - Ο PC αντιγράφεται στον r14_exc.
  - Αλλάζει τον τρόπο λειτουργίας στην αντίστοιχη λειτουργία εξαίρεσης.
  - Ο PC παίρνει την τιμή vector address, οι οποία περιέχει τη διεύθυνση του handler.

Εντολές Επεξεργασίας Δεδομένων (1/2)

- Αριθμητικές λειτουργίες:
  - ADD r0, r1, r2 ; r0 := r1 + r2.
- Λογικές Λειτουργίες:
  - AND r0, r1, r2 ; r0 := r1 AND r2.
- Μετακίνηση Καταχωρητών:
  - MOV r0, r2.
- Σύγκριση:
  - CMP r1, r2.

Εντολές Επεξεργασίας Δεδομένων (2/2)

- Τελεστές:
  - Άμεσοι τελεστές:
    - ADD r3, r3, #1.
  - Τελεστές μητρώου με ολίσθηση:
    - ADD r3, r2, r1, LSL #3.
- Άλλες οδηγίες επεξεργασίας δεδομένων.
  - Πολλαπλασιασμός:
    - MUL r4, r3, r2.

ARM Εντολές δεδομένων

- ADD, ADC : add (w. carry).
- SUB, SBC : subtract (w. carry).
- RSB, RSC : reverse subtract (w. carry).
- MUL, MLA : multiply (and accumulate).

- AND, ORR, EOR.
- BIC : bit clear.
- LSL, LSR : logical shift left/right.
- ASL, ASR : arithmetic shift left/right.
- ROR : rotate right.
- RRX : rotate right extended with C.

Διάφορες Λειτουργίες Δεδομένων

- Λογική ολίσθηση:
  - Συμπληρώνει με 0.
- Αριθμητική ολίσθηση:
  - Συμπληρώνει με 1.
- RRX εκτελεί 33-bit περιστροφή, περιλαμβάνει το C bit από CPSR πάνω μετά το sign bit.

Εντολές για τη μεταφορά δεδομένων

- Φόρτωσε και αποθήκευσε:

LDR r0, [r1].

STR r0, [r1].

  - Offset : LDR r0, [r1, #4]
  - Pre-indexed: LDR r0, [r1], #16
  - Auto-indexed: LDR r0, [r1, #16]!

- Πολλαπλές μεταφορές δεδομένων:
  - LDMIA r1, {r0,r2,r5}.

2 modes: (a) Single Register Transfer (b) Multiple register transfer.

Εντολές για τον έλεγχο της ροής

- Εντολές Διακλάδωσης: B <ετικέτα>.
- Υπό όρους διακλάδωση: BNE <ετικέτα>.
- Διακλάδωση και Σύνδεση: BL <ετικέτα>.

```
        BL loop
        ...  ...
Loop    ...  ...
        ...  ...
        MOV PC,r14  ; επιστροφή
```

ARM εντολές σύγκρισης

- CMP: σύγκριση.
- CMN: αρνητική σύγκριση.
- TST: λογική επεξεργασία δοκιμής.
- TEQ: λογική επεξεργασία αρνητικής δοκιμής.
- Οι οδηγίες αυτές καθορίζουν τα NZCV bits του CPSR.

ARM μετακίνηση εντολών

- MOV, MVN : move (negated):
  - **MOV r0, r1 ; sets r0 to r1.**

ARM οδηγίες load/store

- LDR, LDRH, LDRB: load (half-word, byte).
- STR, STRH, STRB: store (half-word, byte).
- Τρόποι διευθυνσιοδότησης:
  - Άμεσος καταχωρητής: **LDR r0, [r1].**
  - Με δεύτερο καταχωρητή: **LDR r0, [r1,-r2].**
  - Με σταθερά: **LDR r0, [r1,#4].**

ARM ADR ψευδό-πράξεις

- Δεν μπορούν να αναφερθούν σε μια διεύθυνση άμεσα μέσα σε εντολές.
- Δημιουργία της νέας τιμής με την εκτέλεση αριθμητικών πράξεων στο PC.
- Οι ADR ψευδό-πράξεις δημιουργούν εντολές που απαιτούνται για τον υπολογισμό διεύθυνσης:
  - **ADR r1,FOO.**

Παράδειγμα: C αναθέσεις (1/2)

- C:

  x = (a + b) - c;

- Assembler:
  - **ADR r4, a ; get address for a**
  - **LDR r0, [r4] ; get value of a**
  - **ADR r4, b ; get address for b, reusing r4**
  - **LDR r1, [r4] ; get value of b**
  - **ADD r3, r0, r1 ; compute a+b**
  - **ADR r4, c ; get address for c**
  - **LDR r2, [r4] ; get value of c**

C αναθέσεις, συνέχεια…

- SUB r3,r3,r2 ; complete computation of x
- ADR r4,x ; get address for x
- STR r3,[r4] ; store value of x

Παράδειγμα: C αναθέσεις (2/2)

- C:

y = a*(b+c);

- Assembler:
  - **ADR r4,b ; get address for b**
  - **LDR r0,[r4] ; get value of b**
  - **ADR r4,c ; get address for c**
  - **LDR r1,[r4] ; get value of c**
  - **ADD r2,r0,r1 ; compute partial result**
  - **ADR r4,a ; get address for a**
  - **LDR r0,[r4] ; get value of a**

C αναθέσεις συνέχεια..

- **MUL r2,r2,r0 ; compute final value for y**
- **ADR r4,y ; get address for y**
- **STR r2,[r4] ; store y**

Παράδειγμα: C αναθέσεις

- C:

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

- Assembler:
  - **ADR r4,a ; get address for a**
  - **LDR r0,[r4] ; get value of a**
  - **MOV r0,r0,LSL 2 ; perform shift**
  - **ADR r4,b ; get address for b**
  - **LDR r1,[r4] ; get value of b**
  - **AND r1,r1,#15 ; perform AND**
  - **ORR r1,r0,r1 ; perform OR**

C αναθέσεις, συνέχεια (3)

- **ADR r4,z ; get address for z**
- **STR r1,[r4] ; store value for z**

Περισσότεροι τρόποι για την αντιμετώπιση

- Base-plus-offset διευθυνσιοδότησης:
  - **LDR r0,[r1,#16].**
- Φορτώνει από την περιοχή r1+16.
- Auto-indexing προσαυξήσεις βάση καταχωρητή:
  - **LDR r0,[r1,#16]!**
- Post-indexing προσκομίζει, στη συνέχεια, μπορεί να αντισταθμίσει:
  - **LDR r0,[r1],#16.**
- Φορτώνει το r0 από r1, μετά προσθέτει16 στο r1.

ARM ροή του ελέγχου

- Όλες οι λειτουργίες μπορούν να εκτελεστούν υπό όρους, (έλεγχος bit του CPSR):
  - EQ, NE, CS, CC, MI, PL, VS, VC, HI, LS, GE, LT, GT, LE.
- Λειτουργία διακλάδωσης:

  B #100
  - Μπορεί να γίνει υπό όρους.

Εκτέλεση υπό όρους

```
while(i != j) {
    if (i > j)
        i -= j;
    else
        j -= i;
}
```

- αποφεύγει τους κλάδους γύρω από then/else
- If R1=Rj, then neither SUB?? executed

```
loop    CMP     Ri, Rj          ; set condition "NE" if (i != j),
                                ;               "GT" if (i > j),
                                ;            or "LT" if (i < j)
        SUBGT   Ri, Ri, Rj      ; if "GT" (greater than), i = i-j;
        SUBLT   Rj, Rj, Ri      ; if "LT" (less than), j = j-i;
        BNE     loop            ; if "NE" (not equal), then loop
```

Παράδειγμα: δήλωση if

- C:

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

- Assembler:
  - **; compute and test condition**
  - **ADR r4,a ; get address for a**
  - **LDR r0,[r4] ; get value of a**
  - **ADR r4,b ; get address for b**
  - **LDR r1,[r4] ; get value for b**
  - **CMP r0,r1 ; compare a < b**
  - **BGE fblock ; if a >= b, branch to false block**

Παράδειγμα: δήλωση if, συνέχεια.

```
; true block
    MOV r0,#5 ; generate value for x
    ADR r4,x ; get address for x
    STR r0,[r4] ; store x
    ADR r4,c ; get address for c
    LDR r0,[r4] ; get value of c
    ADR r4,d ; get address for d
    LDR r1,[r4] ; get value of d
    ADD r0,r0,r1 ; compute y
    ADR r4,y ; get address for y
    STR r0,[r4] ; store y
    B after ; branch around false block
```

Παράδειγμα: δήλωση if, συνέχεια

- **; false block**
- **fblock ADR r4,c ; get address for c**
- **LDR r0,[r4] ; get value of c**
- **ADR r4,d ; get address for d**
- **LDR r1,[r4] ; get value for d**
- **SUB r0,r0,r1 ; compute a-b**
- **ADR r4,x ; get address for x**
- **STR r0,[r4] ; store value of x**
- **after ...**

Παράδειγμα: υπό όρους εκτέλεση εντολών

```
; true block
— MOVLT r0,#5 ; generate value for x
— ADRLT r4,x ; get address for x
— STRLT r0,[r4] ; store x
— ADRLT r4,c ; get address for c
— LDRLT r0,[r4] ; get value of c
— ADRLT r4,d ; get address for d
— LDRLT r1,[r4] ; get value of d
— ADDLT r0,r0,r1 ; compute y
— ADRLT r4,y ; get address for y
— STRLT r0,[r4] ; store y
```

Παράδειγμα: υπό όρους εκτέλεση εντολών, συνέχεια

; false block

- **ADRGE r4,c ; get address for c**
- **LDRGE r0,[r4] ; get value of c**
- **ADRGE r4,d ; get address for d**
- **LDRGE r1,[r4] ; get value for d**
- **SUBGE r0,r0,r1 ; compute a-b**
- **ADRGE r4,x ; get address for x**
- **STRGE r0,[r4] ; store value of x**

Αλλαγές/περιστροφές στην επεξεργασία δεδομένων

```
a += (j << 2);
```

Θα μπορούσε να αποδοθεί ως εντολή μονολεκτική, ενός κύκλου για την ARM.

```
ADD     Ra, Ra, Rj, LSL #2
```

Παράδειγμα: δήλωση switch

- C:

  switch (test) { case 0: … break; case 1: … }
- Assembler:
  - **ADR r2,test ; get address for test**
  - **LDR r0,[r2] ; load value for test**
  - **ADR r1,switchtab ; load address for switch table**
  - **LDR r1,[r1,r0,LSL #2] ; index switch table**
  - **switchtab DCD case0**
  - **DCD case1**
  - **…**

Παράδειγμα: FIR φίλτρου

- C:

  for (i=0, f=0; i<N; i++)

  f = f + c[i]*x[i];

- Assembler:
  - **; loop initiation code**
  - **MOV r0,#0 ; use r0 for I**
  - **MOV r8,#0 ; use separate index for arrays**
  - **ADR r2,N ; get address for N**
  - **LDR r1,[r2] ; get value of N**
  - **MOV r2,#0 ; use r2 for f**

FIR φίλτρο, συνέχεια

```
ADR r3,c ; load r3 with base of c
ADR r5,x ; load r5 with base of x

; loop body
– loop LDR r4,[r3,r8] ; get c[i]
— LDR r6,[r5,r8] ; get x[i]
— MUL r4,r4,r6 ; compute c[i]*x[i]
— ADD r2,r2,r4 ; add into running sum
— ADD r8,r8,#4 ; add one word offset to array index
— ADD r0,r0,#1 ; add 1 to i
— CMP r0,r1 ; exit?
— BLT loop ; if i < N, continue
```

ARM σύνδεση υπορουτίνων

- Εντολές branch-and-link:
  - **BL foo.**
- **Αντιγραφή του PC στο r14** .
- Για επιστροφή από τη ρουτίνα:
  - MOV r15,r14.
- **Ο παραπάνω μηχανισμός μας αφήνει να καλέσουμε διαδικασίες με βάθος ενός επιπέδου.**
- **Για ένθετες διαδικασίες απαιτείται η κατασκευή στοίβας όπως παρακάτω.**
- **Χρησιμοποιείται ο r13 για να δείχνει την κορυφή της στοίβας.**

Ένθετες κλήσεις υπορουτίνων

- Ένθεση / αναδρομή απαιτεί τη χρήση συγκεκριμένων μοτίβων κωδικοποίησης:
  - **F1 LDR r0,[r13] ; load arg into r0 from stack**
  - **; call f2()**
  - **STR r13!,[r14] ; store f1's return adrs**
  - **STR r13!,[r0] ; store arg to f2 on stack**
  - **BL f2 ; branch and link to f2**
  - **; return from f1()**
  - **SUB r13,#4 ; pop f2's arg off stack**
  - **LDR r13!,r15 ; restore register and return**

Περίληψη

- Οι ARM είναι Load/store αρχιτεκτονική.
- Οι περισσότερες οδηγίες είναι RISC, λειτουργούν σε ενιαίο κύκλο.
- Μερικές σύνθετες εντολές διαρκούν περισσότερο.
- Όλες οι οδηγίες μπορούν να εκτελεστούν υπό όρους.

Αναφορές

- Χρησιμοποιήθηκε υλικό από παρουσιάσεις των:
  - Dimitrios Soudris, NTUA, 2012.
  - Wayne Wolf, “Computers as Components” Overheads, 2008.
  - Leonid Ryzhyk, “The ARM Architecture”, 2006.

Τέλος Ενότητας