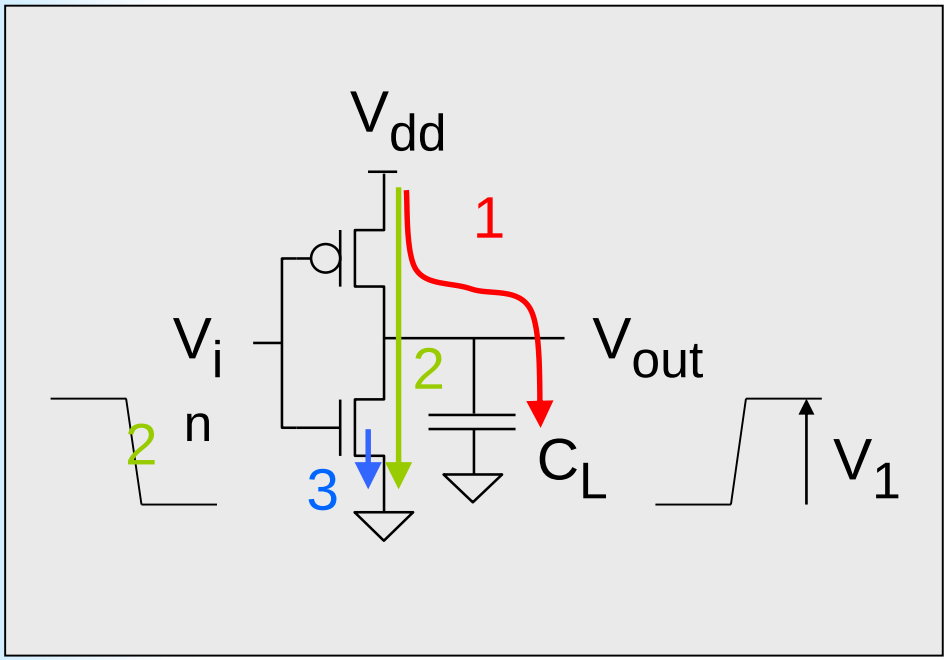
Thumb2: A Thumb Extension to The ARM Architecture

Peng-Sheng Chen

Fall, 2017

先來看看 Power (電能消耗)

CMOS Power Components



1. Dynamic/Active power (Switching power)
   Charging and discharging capacitors
2. Static/Leakage power
   Reverse biased diodes
3. "Short Circuit" power
   During switching direct path VDD-GND

Switching power
$= \frac{1}{2}\ CL\ V2DD$
$\cong 1$ picojoule

Power consumption

Power consumption = Static power + Dynamic power

Where

Static Power = Σ leakage current * V_{DD}

Σ: Summary of number of devices

Dynamic Power = $C_L V^2_{DD} f$

Where C_L is the capacitance loading,

V_{DD} is the power supply

f is the running frequency.

系統設計的問題

System Design

- **Design issues**
  - Performance
  - Code density (code size)：完成一件工作，所需要的程式碼的大小
    - Reduce memory footprints
  - Power consumption

Design Trade-Offs (1)

- **Code density**
  - Memory size
  - Memory is one of major components of system costs
  - Fewer memory footprints => higher performance
  - Fewer memory footprints => lower power consumption
  - Thumb instruction set
    - Possibly need more instructions to represent a 32-bit ARM instruction => bad performance

Design Trade-Offs (2)

- **Performance**
  - Fewer instructions => better performance
  - Fewer memory footprints => better performance
  - Lower clock speed => reduce power consumption, bad performance

Design Trade-Offs (3)

- **Power consumption**
  - Low voltage, low frequency => low power (bad performance)
  - Fewer memory footprints => low power
  - Off-chip data / instruction transactions will tend to dissipate more power
    - The size of on-chip memory is limited => code density

System Design

- **Design issues**
  - Performance
  - Code density (code size)：完成一件工作，所需要的程式碼的大小
    - Reduce memory footprints
  - Power consumption

如何取得平衡，很複雜，系統開發時很難評估

Solution for Design Trade-Offs

- Using a **combination of ARM and Thumb code** within an application enables designers to **balance**
  - Code density
  - Performance
  - Power characteristics

Drawback

- You need to switch between ARM instructions and Thumb instructions
- Thumb instruction set is a subset of ARM instructions
  - **Cannot access special functions, ex: SIMD**
  - **Cannot access coprocessor**
  - **Cannot access privileged instructions**

Thumb2 instruction set

Thumb2 Instruction Set (1)

- The Thumb2 ISA consists of the existing 16-bit Thumb instructions augmented by
  - New 16-bit Thumb instructions for improved program flow
  - New **32-bit Thumb instructions** derived from ARM instruction equivalents

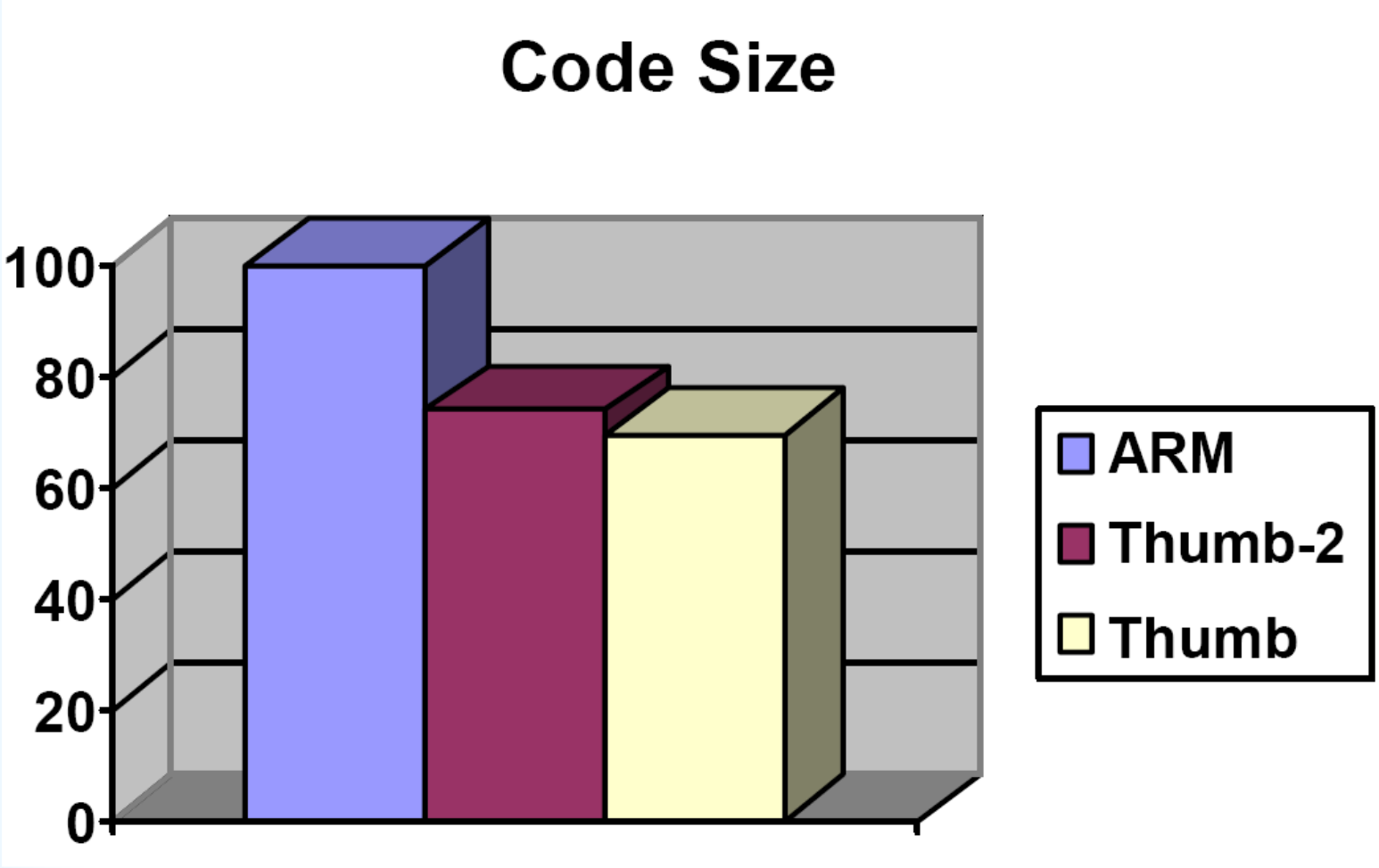
Thumb2 Instruction Set (2)

- The addition of 32-bit instructions to Thumb overcomes the previous limitations
  - Co-processor access
  - Privileged instructions
  - Special function instructions, ex: SIMD
- **Thumb can now access all of the instructions it needs to enable both high performance and exceptional code density**

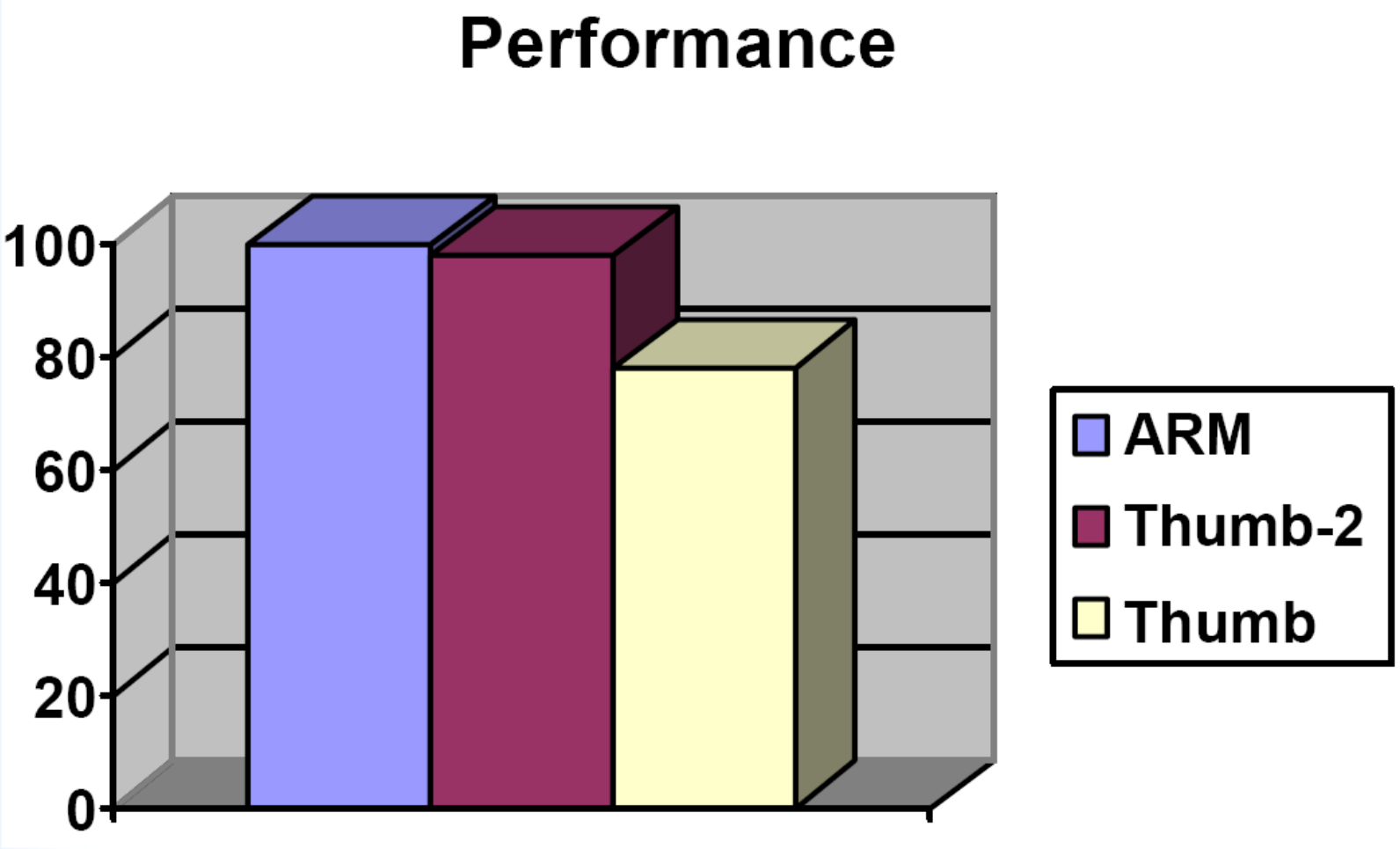
Thumb2 Instruction Set (3)

- **The new Thumb-2 core technology provides**
  - Access to equivalents for virtually all ARM instructions
  - 12 completely new instructions which change the performance-code size balance
  - Performance which reaches 98 percent of C code developed using the ARM instruction set alone
  - Memory footprint which is just 74 percent of an ARM–equivalent implementation

The Comparison on Code Size

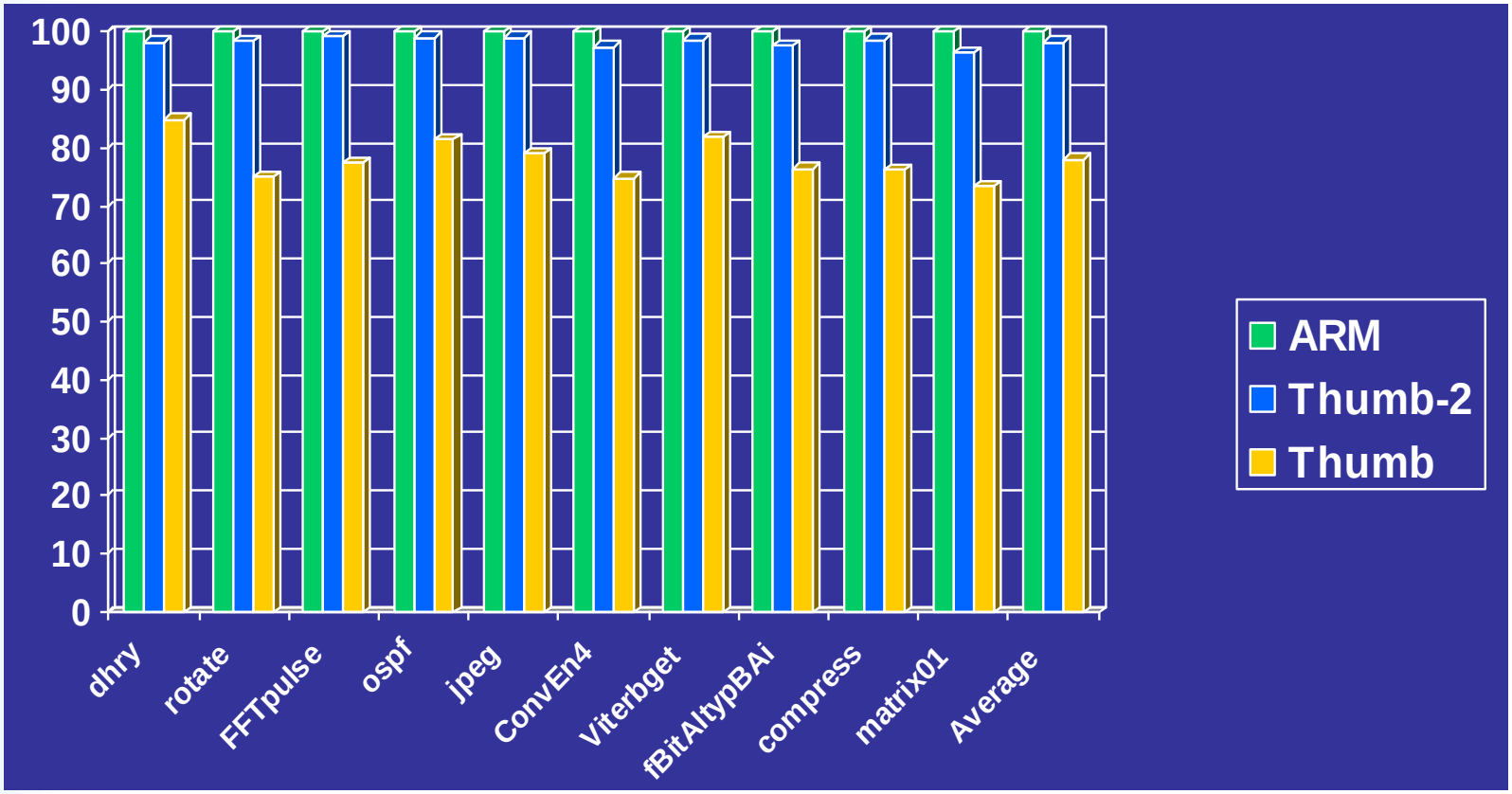


The Comparison on Performance



Thumb-2 Performance

Analysis of the performance of code for EEMBC* benchmarks on ARM11 like cores

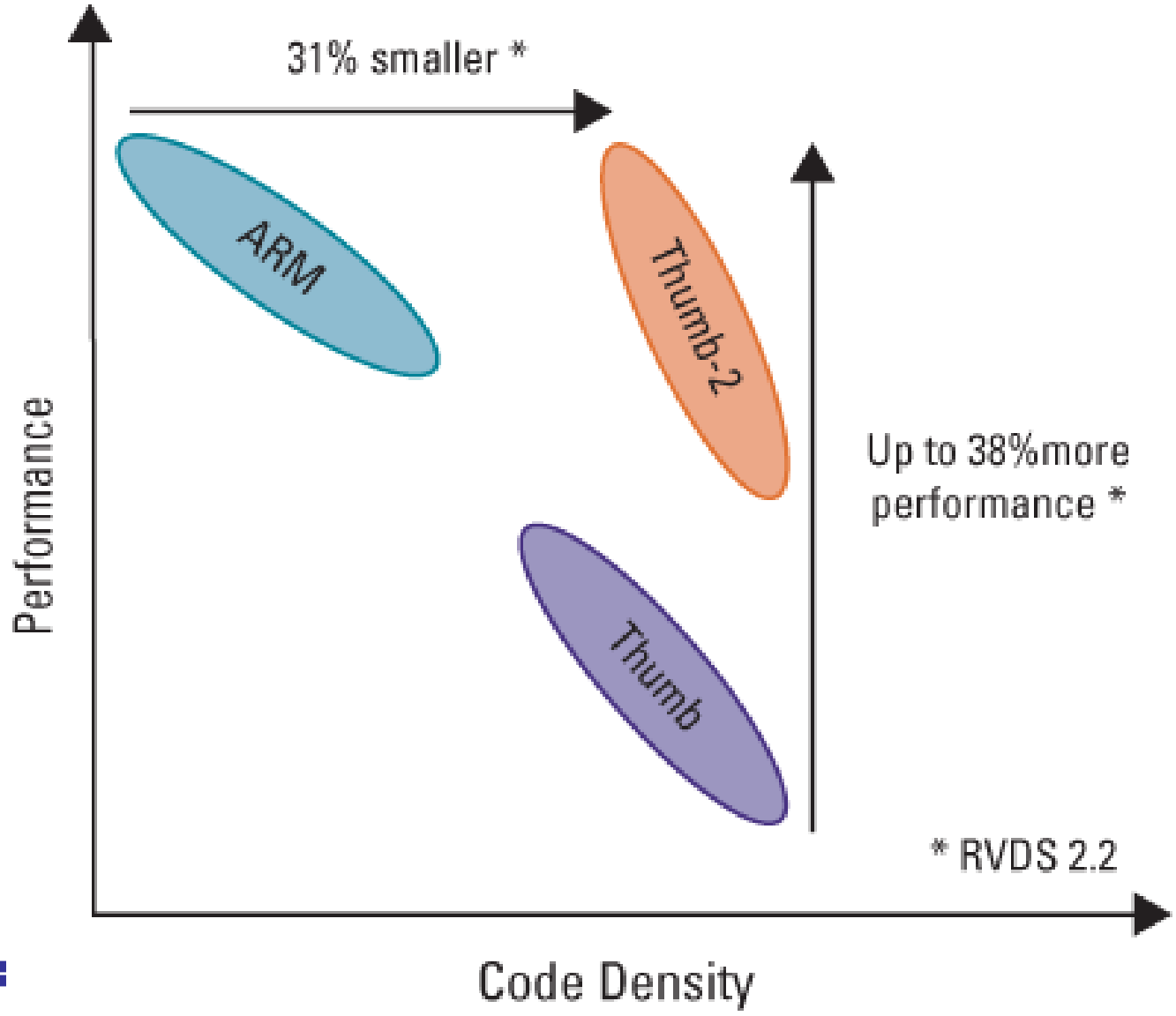


Thumb-2 performance is 98% of ARM performance

Thumb-2 code achieves 125% of Thumb performance

Reference from: http://www.arm.com

The Trend


31% smaller *
ARM
Thumb-2
Thumb
Up to 38%more
performance *
* RVDS 2.2
Performance
Code Density

Benefits

- Mixing between ARM and Thumb instructions will no longer be required
- Balance performance and code size
- Simply design process

Applications

- 汽車的行車電腦
  - Anti-lock braking system (ABS)
  - 車輛穩定系統
- 隨著記憶體在晶片中所佔比重愈來愈高，運用新技術 – Thumb-2 – 開發汽車產品的晶片製造商，必能節省大量的空間與成本

Thumb2 32-bit Instructions

ARM-like:

- Data Processing instructions
- DSP and Media instructions
- Load and Store instructions
- Branch instructions
- System control – BXJ, RFE, SRS etc.
- Coprocessor (VFP, MOVE™, etc.)

New:

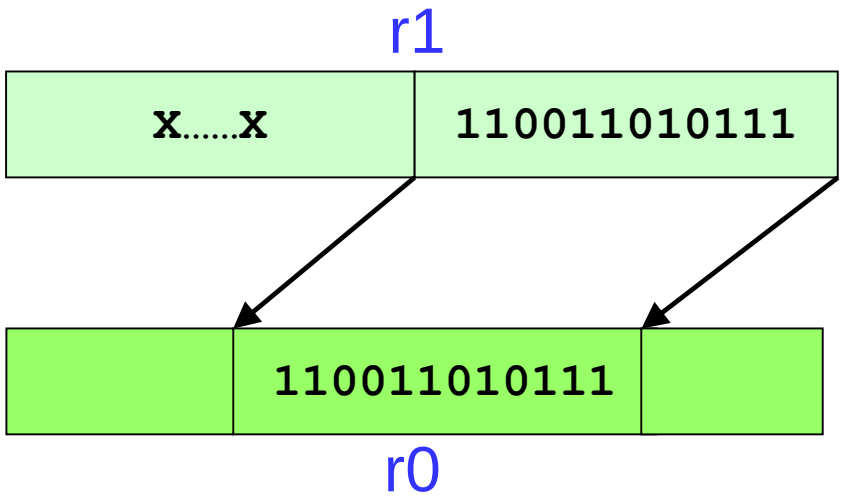
- Bitfield insert/extract/clear BFI, {S|U}BFX, BFC
- Bit reverse RBIT
- 16 bit immediate instructions MOVW, MOVH
- Table branch TB{B|H} [Rbase, Rindex]
- Additional memory system hints (PLD)

Bit-Field Instructions (1)

- 把暫存器Rx的值放入暫存器Ry的特定位置中
- Ex: 把r1的12位元資料插入到r0的4~15 bit

ARM (v6 or earlier)

```
; pack r1 into r0
; mask = 0x00000FFF
AND     r1, r1, mask
BIC     r0, r0, mask, LSL #4
ORR     r0, r0, r1   , LSL #4
```



Bit-Field Instructions (2)

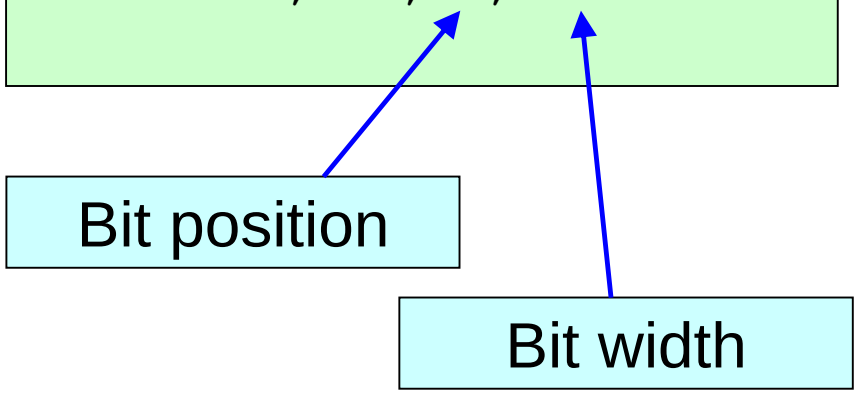
- 把暫存器Rx的值放入暫存器Ry的特定位置中
- Ex: 把r1的12位元資料插入到r0的4~15 bit

ARM (v6 or earlier)

```
; pack r1 into r0
; mask = 0x00000FFF
AND     r1, r1, mask
BIC     r0, r0, mask, LSL #4
ORR     r0, r0, r1  , LSL #4
```

Thumb2 (Thumb or ARM)

```
; pack r1 into r0
; mask = 0x00000FFF
BFI     r0, r1, 4, 12
```



New Instructions: IT (If …Then)

- It's better to have conditional execution for Thumb code
- The IT instruction predicates the execution of up to **four Thumb instructions** on the basis of one of the condition codes

ARM

```
LDREQ r0,[r1]
LDRNE r0,[r2]
ADDEQ r0, r3, r0
ADDNE r0, r4, r0
```

16 bytes

Thumb

```
 BNE L1
 LDR r0, [r1]
 ADD r0, r3, r0
 B L2
L1
 LDR r0, [r2]
 ADD r0, r4, r0
L2
```

12 bytes

Thumb-2

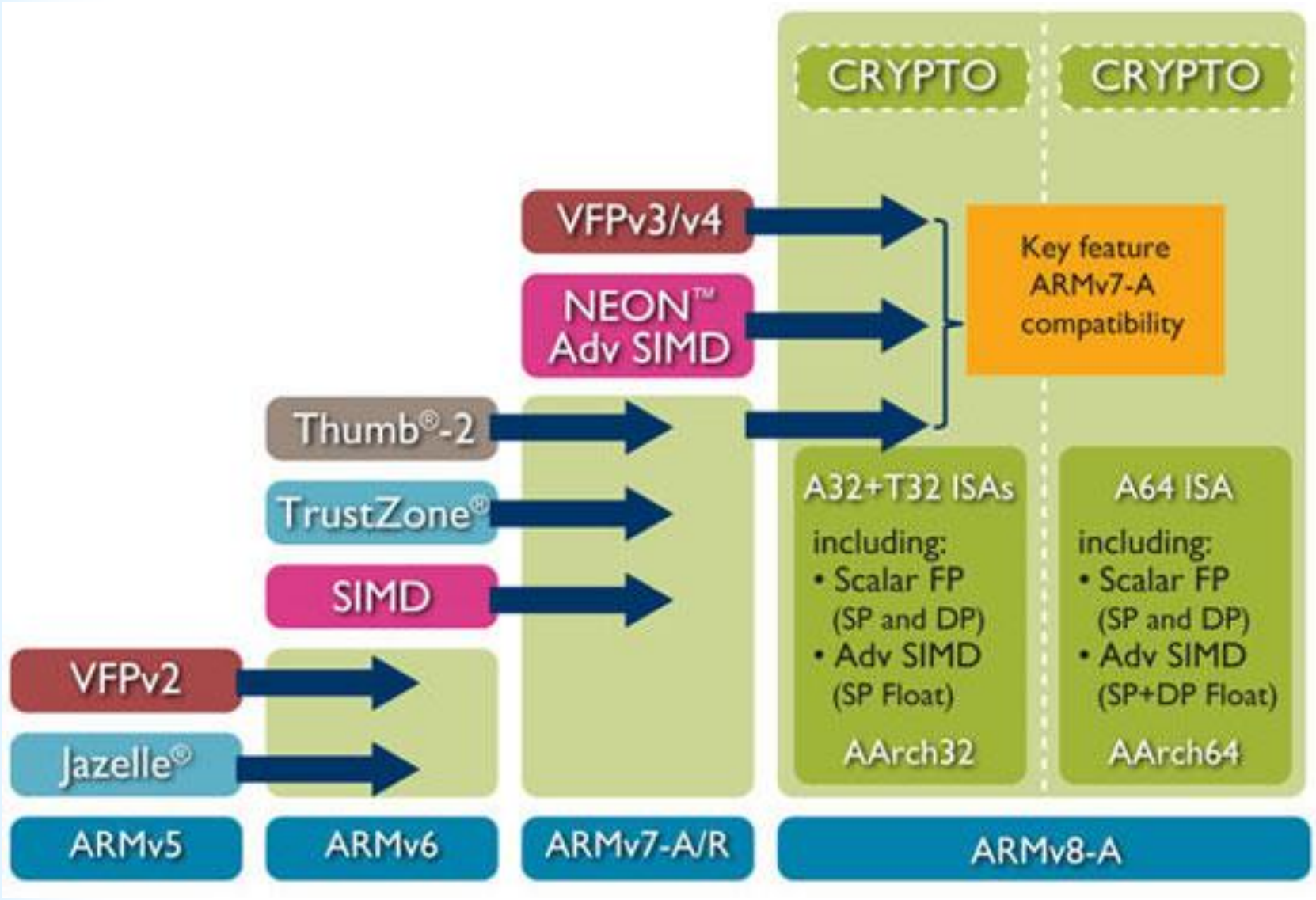
```
ITETE EQ
LDR r0, [r1]
LDR r0, [r2]
ADD r0, r3, r0
ADD r0, r4, r0
```

10 bytes

Case Study: Application Example

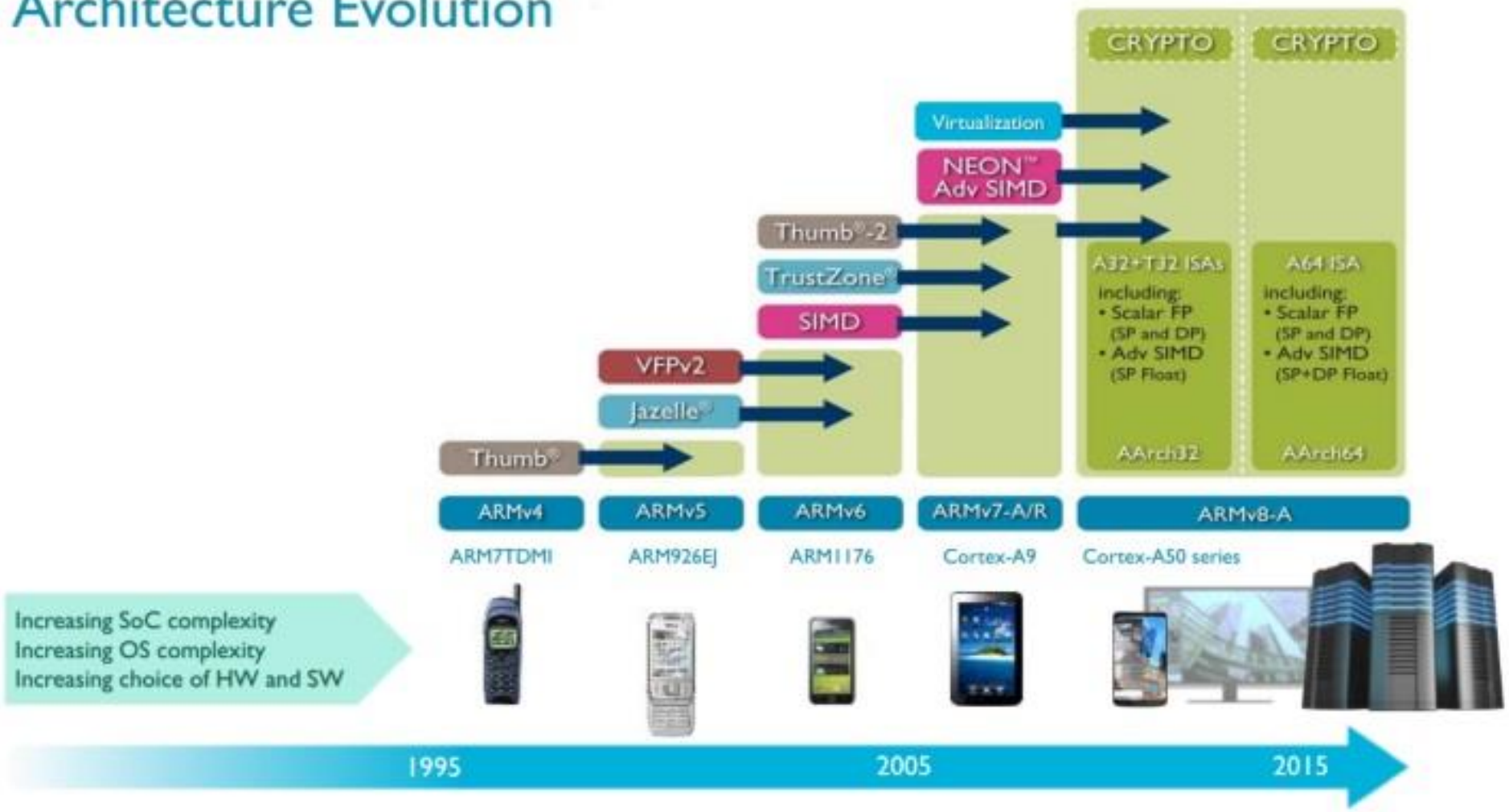
- The 1MB application was originally implemented with
  - 80 percent of the code compiled for Thumb to achieve the best code density
  - 20 percent targeting the ARM instruction set to get the required performance

ARM	Thumb-2
200k	150k
Thumb	**Thumb-2**
800k	760k



http://www.arm.com/products/processors/instruction-set-architectures/index.php

Architecture Evolution



Reference from: http://www.androidauthority.com/developing-arm-everything-need-know-389402/

Summary

- Thumb-2 core technology provides performance at higher code densities than previously achievable with the ARM architecture
- Thumb-2 introduces a number of new features to further improve program flow, program efficiency, and code size
- Together these benefits will enable designers to pack more features into devices while obtaining improved power and performance characteristics

Reference

- "Improving ARM Code Density and Performance: New Thumb Extensions to the ARM Architecture", Richard Phelan June 2003 (http://www.arm.com)