Instruction Set Architectures (ISAs)

Simon Hollis, COMS12200

Example ISA

The ARM Thumb V1 ISA

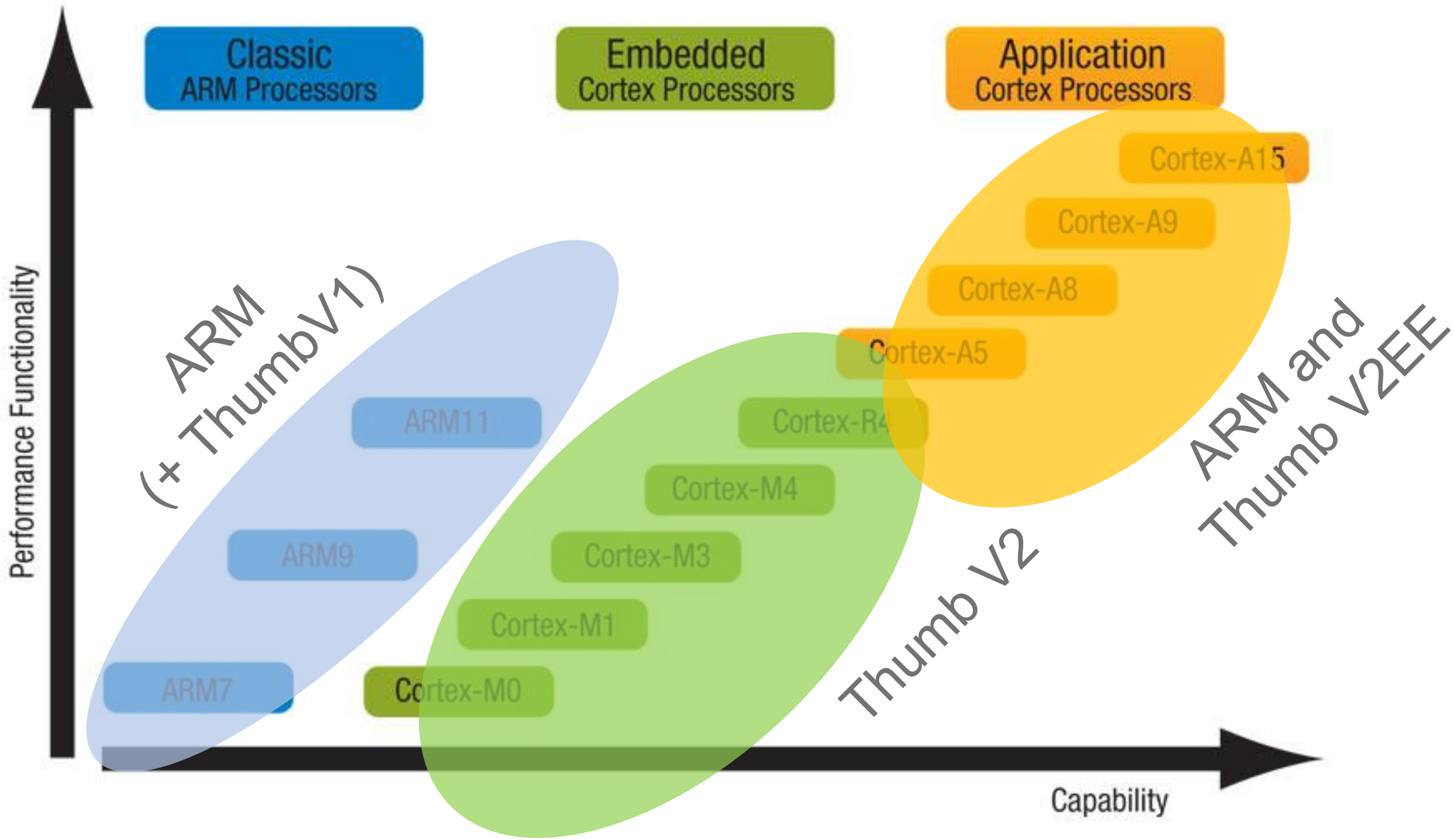
A concrete ISA

- So far, we have talked about ISAs in a generic manner.
- In this lecture and the following one, we will discuss a concrete ISA: the ARM Thumb V1 ISA.

Why Thumb V1?

- ARM is the world's most populous processor
  - Billions shipped per year
- They have multiple device families supporting various ISAs.
  - ARM (32-bit)
  - Thumb V1 (16-bit)
  - Thumb V2 (16-bit and 32-bit)
  - Thumb V2EE (16-bit and 32-bit)

ARM device families



Why Thumb V1?

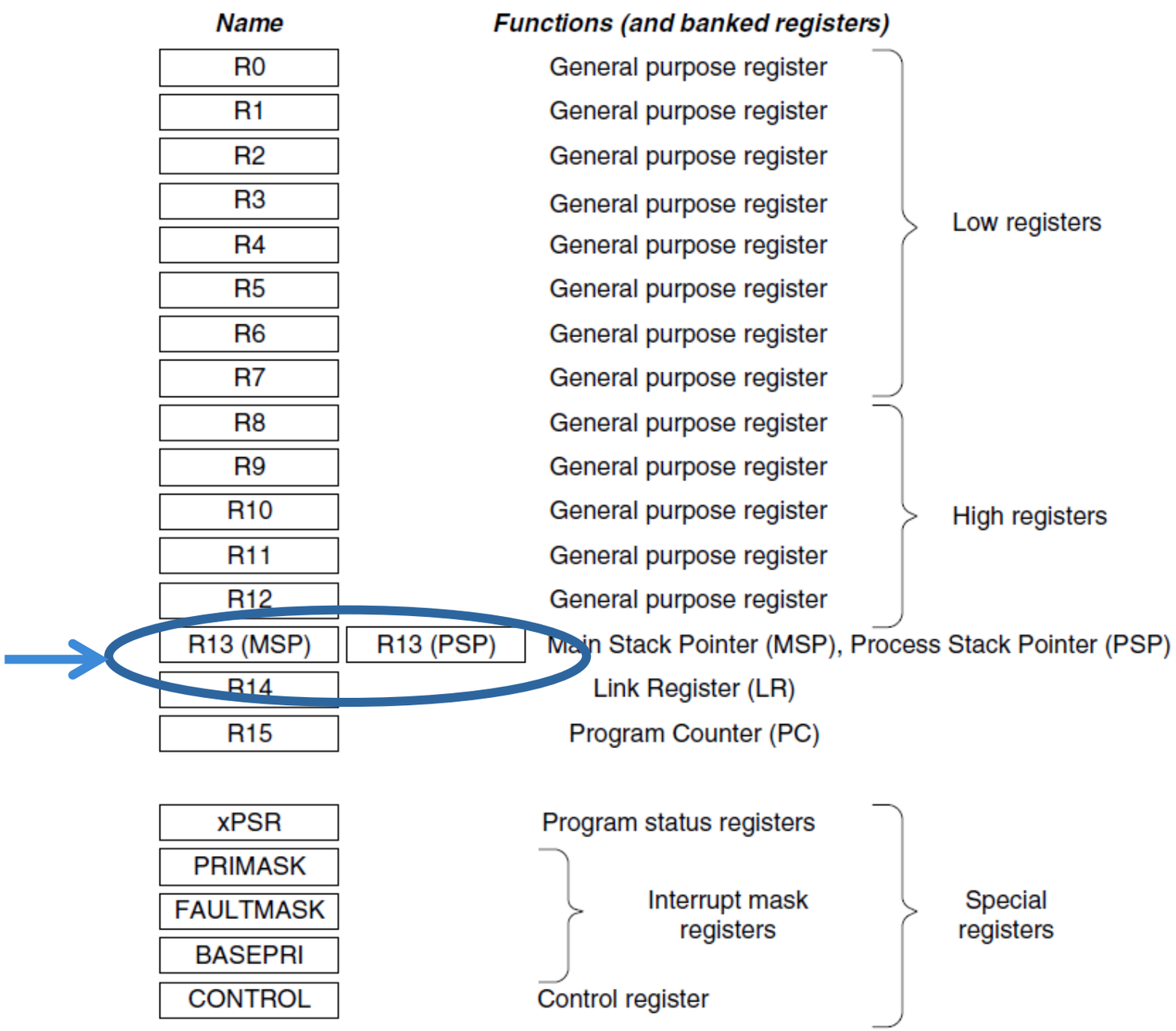
- Thumb V1 is powerful and compact.
- It is contemporary and delivers good performance for its code size.
- Most importantly for this course, the instructions contained within are relatively simple and few in number.
- All Thumb V1 is also valid Thumb V2.

The ARM architecture

- Before we look a the Thumb ISA, let's look at the ARM architecture at a high-level.
- ARM:
  - Is a register machine, with 16 registers.
    - The PC and LR are part of these registers
  - Is RISC.
  - Has good support for multiple memory addressing modes and is effective with stacks.

ARM M-series registers

Note the stack pointer register is duplicated (more in COMS35102!)



Yiu: Definitive Guide to the ARM Cortex M3

FIGURE 3.1

Registers in the Cortex-M3.

Flexible registers

- The orthogonality of the ARM registers means that *any* combination of them can be used for the source and destination of *any* instruction.
- This gives great flexibility in
  - Register assignment
  - Variable manipulation
- It *includes the Program Counter*
  - Branches can be made as the result of an arithmetic calculation!

Memory manipulation

Most of the addressing modes we saw in previous lectures are supported by the ARM architecture, including

- Register-register
- Register-memory
- Immediate
- Indexed

Memory manipulation

In general, memory addresses are accessed in a two phase approach.

1. Store the memory address to be accessed in a register.
2. Execute a load or store instruction, targeting that register's address.

Memory access example

e.g. to write ‘42’ to memory location 8.

```
1. MOV r0, #8     ; place 8 in r0
2. MOV r1, #42    ; place 42 in r1
3. STR r1, [r0]   ; write value in r1 to
                    address in r0
```

ARM PC

- The ARM's program counter is pre-incremented in an execution cycle.
- It runs ahead of the current instruction by 2 instructions.
  - This is because it follows the three stage pipeline model we saw before.

Thumb V1 overview

- The Thumb V1 is a 16-bit instruction set.
  - This places some limitations on what it can and cannot do.
    - E.g. encoding three registers from the full ARM set takes 12 bits.
    - There cannot be very many op-codes
      - Op-code length is variable at 5—10 bits

Thumb V1 overview

- Instructions are broadly split into two forms:
  1. Three register operations
     - E.g. ADD r0, r1, r2
  2. Two register + one immediate operations
     - E.g. ADD r0, r1, #10

Thumb V1 overview

- Most Thumb V1 instructions address one of the 8 ‘low’ registers (i.e. `r0–r7`).
- There are two main forms of register encoding:
  - Three explicit registers
    - e.g. ADD r0, r1, r2
  - Two explicit + one implicit
    - e.g. AND r0, r1

‘High’ instructions

- Some instructions that need to access registers `>r7` do allow access to the full register set.
- These instruction encodings are known as ‘high’ instructions.
  - e.g. ADDH, MOVH
  - Often some other functionality of the instruction is sacrificed, such as 3 register format being reduced to 2 register format

Conditionality

- Conditionality in the ARM family is powerful and complex in equal measure.
- Conditional execution occurs in the following way:

1. Certain instructions' execution has a side-effect of setting conditional bits in the status register.
2. Subsequent instructions can become conditional on the basis of these set bits.

Thumb conditional example

Here's an example:

```
1. ADD r0, #2    ; r0 = r0 + 2

2. BEQ loop  ; branch to loop if
                 previous result was
                 zero. (i.e. r0 started as -2)
```

Thumb conditional

The conditional information was stored in a hardware element called the *status register*.

Here's some important parts:

Bit 31	Bit 30	Bit 29	Bit 28
N	Z	C	V
Negative	Zero (Equal)	Carry	Overflow

Thumb control flow

- Control flow for the Thumb is fairly standard
  - Unconditional branches (BU)
  - Conditional branches (BC)
  - Calls (BL)

- As expected, control flow changes cause a pipeline flush and a execution time penalty.

Thumb control flow

- However, the limited instruction width places restrictions on the reachable distances.
  - BU: 11 bits of immediate
  - BC: 8 bits of immediate
  - BL: 21 + 3 bits of immediate

Double-length `BL`

- The “Branch with Link” (`BL`) instruction is unusual.
  - It needs greater than 16 bits to encode the jump address.
  - So, it is formed of two separate 16 bit instructions, that are concatenated in hardware.
- The first instruction is called “BL1”, the second “BL2”.
  - Both are necessary for a successful operation.

ARM stack model

ARM stacks are full, descending

Address	Stack pointer order
0x2010000	0 (initial for stack starting at 0x200ffff)
0x200fffc	1
0x200fff8	2
0x200fff4	3
0x200ffe0	4
0x200ffec	5
0x200ffe8	6
0x200ffe4	7
….	….

PUSH and POP

The ARM architecture has plenty of support for stack based operation.

`PUSH` and `POP` are dedicated instructions to operate on a stack.

- Stack location is defined using the stack pointer (=r13).

PUSH and POP example

This sequence of operations:

```
1. MOV r0, #2
2. MOV r1, #15
3. MOV r2, #31
4. PUSH r0
5. PUSH r1
6. POP r0
7. PUSH r2
8. PUSH r0
9. POP r0
```

registers

r0	r1	r2

memory

Stack pointer	Address	Data value
	0x200fffc	
	0x200fff8	
	0x200fff4	
	0x200fff0	
	0x200ffec	
	0x200ffe0	

Thumb V1 quirks

There are certainly some quirks in the Thumb V1 ISA.

- Multiple op-codes for some operations
- Overlapping prefixes
- Hard to reach registers

You'll get to grips with these in the coursework ☺

Summary of Thumb V1

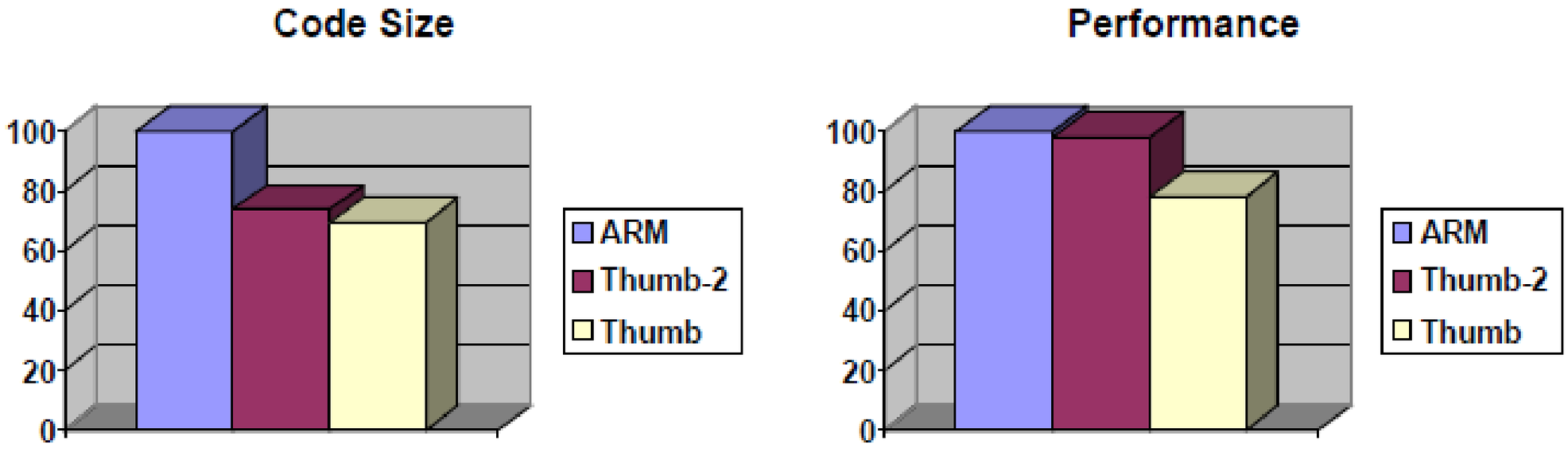
- Thumb V1 is a 16-bit instruction set.
- It predominantly targets only 8 of the 16 registers.
- Conditionality is a combination:
  - Instructions that implicitly set status bits
  - Instructions that are explicitly conditional
- Control flow suffers from restrictions.
- It is, however, compact and efficient.

A word about Thumb V2

- Thumb V2 is newer than V1.
- Many newer ARM processors support it.
- It includes additional instruction formats and features.
- It is much more complex with more variable length instructions.
- The additional complexity is a big disadvantage.
- It does offer some performance benefits over V1.

Relative efficiency

- Thumb v2 benchmarks
  - 74% memory footprint c.f. ARM
  - 98% performance c.f. ARM



Source: http://www.arm.com/pdfs/Thumb-2CoreTechnologyWhitepaper-Final4.pdf