

Introdução à arquitetura AArch64

João Canas Ferreira

Março 2019



Assuntos

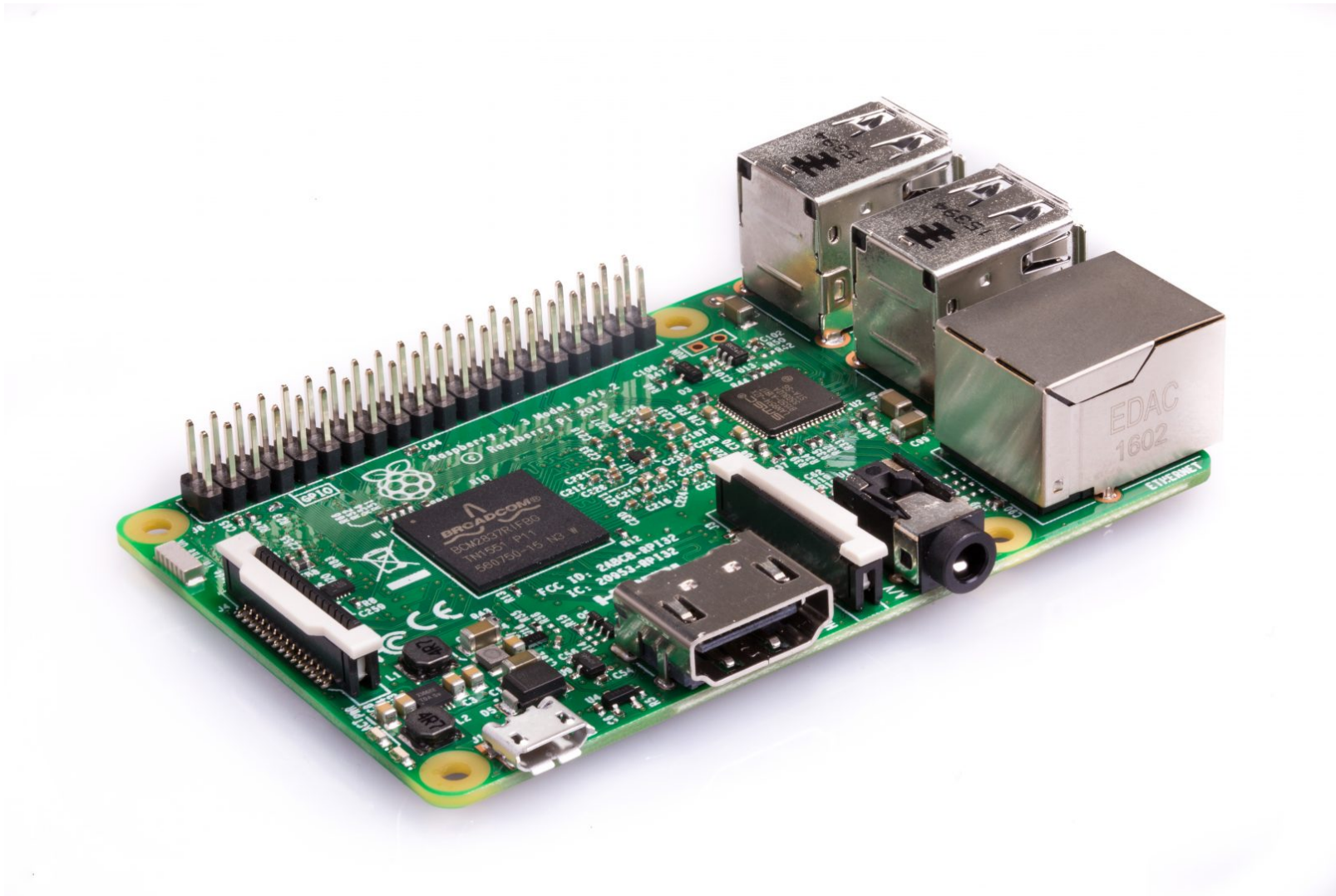
1 Microprocessadores ARM

2 Instruções básicas

As arquiteturas ARM

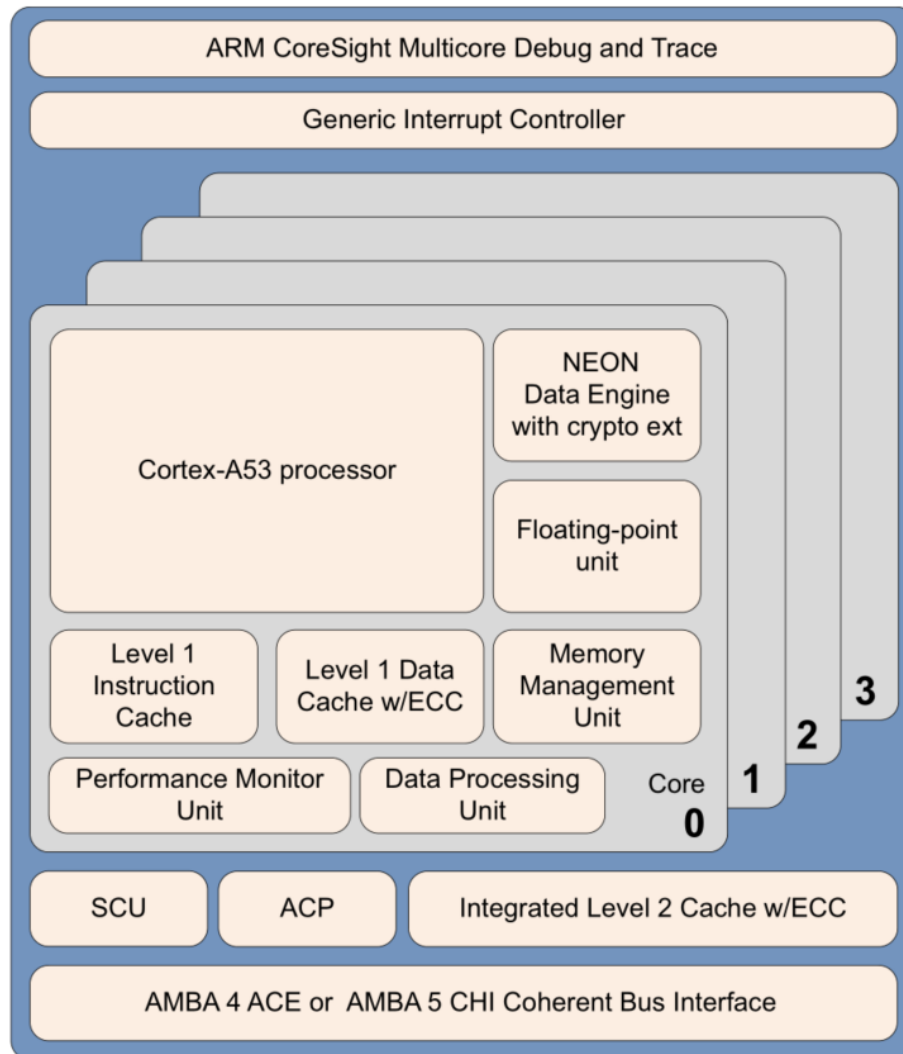
- A designação ARM cobre uma família de arquiteturas RISC.
- Arquiteturas predominantes em *smartphones*, *tablets* e dispositivos de IoT (Internet of Things)
- A companhia ARM licencia a sua propriedade intelectual a outras companhias (Samsung, NXP, etc.).
O Raspberry PI 3 usa um circuito BCM2837 da companhia Broadcom, que inclui um CPU Cortex-A53 com 4 núcleos.
- Arquitetura ARMv8 tem dois modos de execução:
 - AArch64** Execução de aplicações de 64 bits
 - AArch32** Execução de aplicações de 32 bits, compatível com ARMv7-A.
- Vamos tratar a arquitetura do conjunto de instruções **AArch64**.


Raspberry PI 3



Dimensões: 85 mm×56 mm

Cortex-A53



- 4 núcleos
- unidade de vírgula flutuante
- instruções SIMD (NEON) 
- controlador de interrupções
- memória *cache* L1 (por núcleo) e L2 (comum)
- ECC: código corretor de erros
- SCU, ACP: módulos de apoio (*cache*)
- AMBA: barramento interno

Perfis (variantes)

- Existem três variantes (perfis) da arquitetura ARM para mercados/aplicações diferentes.

A — Application Suporta memória virtual via uma MMU (*Memory Management Unit*); usada com sistemas operativos como Linux, iOS, Android.

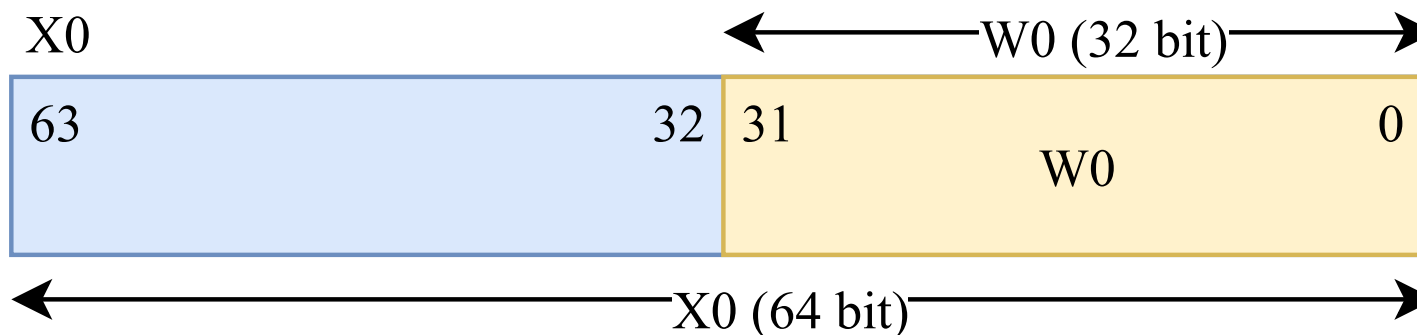
R — Real-time Para aplicações de tempo real (aplicações médicas, automóveis, aviação, robótica); baixa latência e elevado nível de segurança.

M — Microcontroller Proteção de memória; usado para gestão de energia, entradas/saídas, ecrãs táteis, controladores de sensores.

- Nesta u.c focamos o perfil A, que é usado na maioria dos SBC (*Single-Board Computer*).
- ARM está a entrar no mercado de HPC (*High-Performance Computing*) (servidores e computação científica).

AArch64: Modelo de programação (1/3)

- Arquitetura do conjunto de instruções (ISA) **ortogonal** do tipo *load/store*; endereços de 64 bits.
- Todas as instruções têm o mesmo comprimento: 32 bits.
- 31 registos de uso geral: X0–X30.
- 1 registo “virtual” com dupla função: tem o valor 0 ou o **apontador para topo da pilha**.
- O *program counter* não é diretamente acessível.
- Os 32 bits menos significativos de cada registo de uso geral são designados por W0–W30.



AArch64: Modelo de programação (2/3)

- Geralmente, as instruções têm 3 operandos: destino, fonte1 e fonte2
- Os operandos podem ser de 32 ou 64 bits. Exemplo:

```
ADD    W0,W1,W2    // adição de registos de 32 bits
ADD    X0,X1,X2    // adição de registos de 64 bits
ADD    X0,X1,#42   // adição de um valor imediato a registo de 64 bits
```
- Atribuições de um valor a registos W<n> colocam os bits mais significativos de X<n> a zero.
- Na maioria dos casos, o registo 31 produz o valor 0 e não é alterado na escrita.
- Quando usado como registo de endereço em instruções *load/store* e em algumas operações aritméticas, o registo 31 dá acesso ao **apontador para o topo da pilha**.
- São permitidos acessos não-alinhados a memória num grande número de situações. Convenções de organização de código podem impor condições de alinhamento (software).

AArch64: Modelo de programação (3/3)

► Existem quatro indicadores (*flags*) NZCV com o seguinte significado:

Flag	Nome	Descrição
N	Negativo	N = bit mais significativo do resultado (1→negativo)
Z	Zero	Z=1 se valor igual a 0
C	Carry	Fica a 1 se a operação faz <i>overflow</i> (sem sinal)
V	Overflow	Fica a 1 se a operação faz <i>overflow</i> (com sinal)

► Códigos de condições <cond> (para instruções condicionais):

Cód.	Significado	Cond.	Cód.	Significado	Cond.
EQ	igual a	Z=1	NE	diferente de	Z=0
CS	carry set	C=1	HS	igual ou maior (s/ sinal)	C=1
CC	carry clear	C=0	LO	menor que (s/ sinal)	C=0
MI	negativo	N=1	PL	positivo ou zero	N=1
VS	overflow (c/ sinal)	V=1	VC	sem overflow (c/ sinal)	V=0
HI	maior que (s/ sinal)	(C=1) e (Z=0)	LS	menor ou igual (s/ sinal)	(C=0) e (Z=1)
GE	maior ou igual (c/ sinal)	N=V	LT	menor que (c/sinal)	N!=V
GT	maior que (c/ sinal)	(Z=0) e (N=V)	LE	menor ou igual (c/sinal)	(Z=1) e (N!=V)

AL ou NV: execução sem condições

Assuntos

1 Microprocessadores ARM

2 Instruções básicas

Instruções de processamento de dados


▢ Estas instruções têm o formato: **Instrução Rd, Rn, Op2**

Rd Registo de destino (X0, X1, ..., X30 ou W0, W1, ..., W30)

Rn Fonte de um dos valores usados na operação

Op2 Fonte do outro valor usada na operação.

▢ **Op2** pode ser um registo, um registo *modificado* ou um valor imediato.

Tipo	Instruções
Aritmética	ADD{S}, SUB{S}, ADC{S}, SBC{S}, NEG
Lógicas	AND{S}, BIC{S}, ORR, ORN, EOR, EON 
Comparação	CMP, CMN, TST
Movimento	MOV, MOVN

▢ As variantes que terminam com S ({s}) afetam as *flags*.

▢ As instruções de comparação também afetam as *flags*.

▢ Valores imediatos têm 12 bits.

Instruções de deslocamento

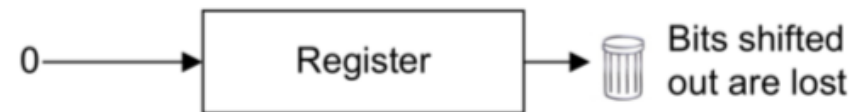
► Instruções LSL, LSR, ASR e ROR

LSL Logical shift left



Multiplication by 2^n where n is the shift amount

LSR Logical shift right



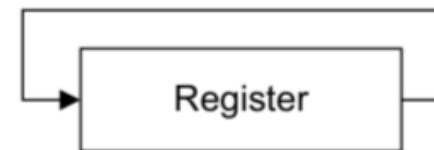
Unsigned division by 2^n where n is the shift amount

ASR Arithmetic shift right



Division by 2^n , where n is the shift amount, preserving the sign bit

ROR Rotate right



Bit rotate with wrap around from LSB to MSB

Controlo de fluxo

➡ Instruções de salto / chamada de sub-rotina / retorno de sub-rotina

Instrução	Efeito
B etiqueta	salto para $PC \pm 128 \text{ MiB}$
BL etiqueta	como B, mas guarda endereço de retorno em X30
B.<cond> etiqueta	salto condicional para $PC \pm 1 \text{ MiB}$
BR X<n>	salto para posição com endereço no registo X<n>
BLR X<n>	como BR, mas guarda endereço de retorno em X30
RET {X<n>}	como BR, mas indica retorno de função; por omissão n=30

➡ Instruções de salto condicional com comparação incluída

Instrução	Efeito
CBZ Reg, etiqueta	se Reg=0, salto para $PC \pm 1 \text{ MiB}$
CBNZ Reg, etiqueta	se Reg=1, salto para $PC \pm 1 \text{ MiB}$
TBNZ Reg, bit, etiqueta	se Reg[bit]=0, , salto para $PC \pm 32 \text{ KiB}$
TBNZ Reg, bit, etiqueta	se Reg[bit]=1, salto para $PC \pm 32 \text{ KiB}$

Reg pode ser X<n> ou W<n>

CB: Compare and branch TB: test and branch

Instruções de leitura de memória

Formato: **LDR Reg, endereço**

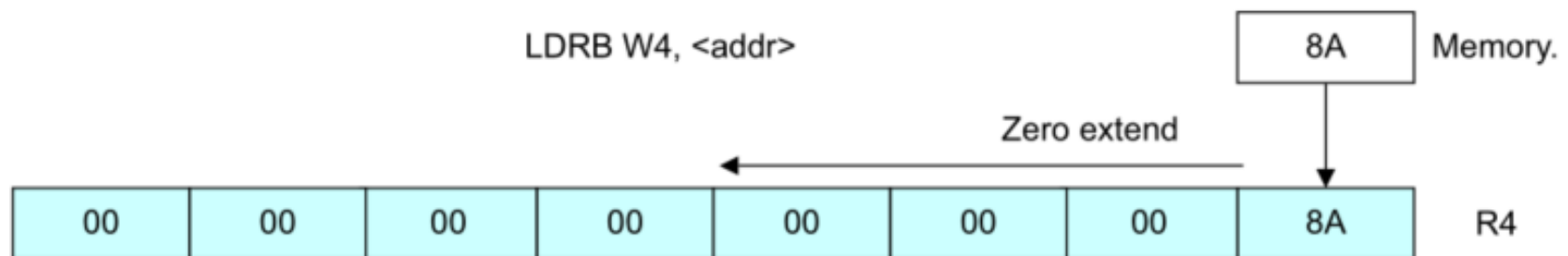
Instrução	Tamanho	Extensão	Registo
LDRB	8 bit	com 0	W<n>
LDRSB	8 bit	de sinal	W<n> ou X<n>
LDRH	16 bit	com 0	W<n>
LDRSH	16 bit	de sinal	W<n> ou X<n>
LDRSW	32 bit	de sinal	X<n>

Formato de endereço (simples)

Exemplo	Descrição
LDR X0, [X1]	endereço dos dados é o valor de X1
LDR X0, [X1, #8]	endereço dos dados é o valor de X1+8
LDR X0, [X1,X2]	endereço dos dados é o valor de X1+X2

Endereço é sempre um número de 64 bits.

Leitura com extensão de representação



Instruções de escrita em memória

Formato: **STR Reg, endereço**

As instruções seguintes escrevem em memória a parte menos significativa de um registo W<n>.

Instrução	Tamanho
STRB	8 bit
STRH	16 bit

Formato de endereço é o mesmo que é usado com as instruções de escrita.

Para valores de 32 ou 64 bits, usar os registos respetivos. Exemplos:

- 32 bits: STR W0, [X1, #8]
- 64 bits: STR X0, [X1, #8]