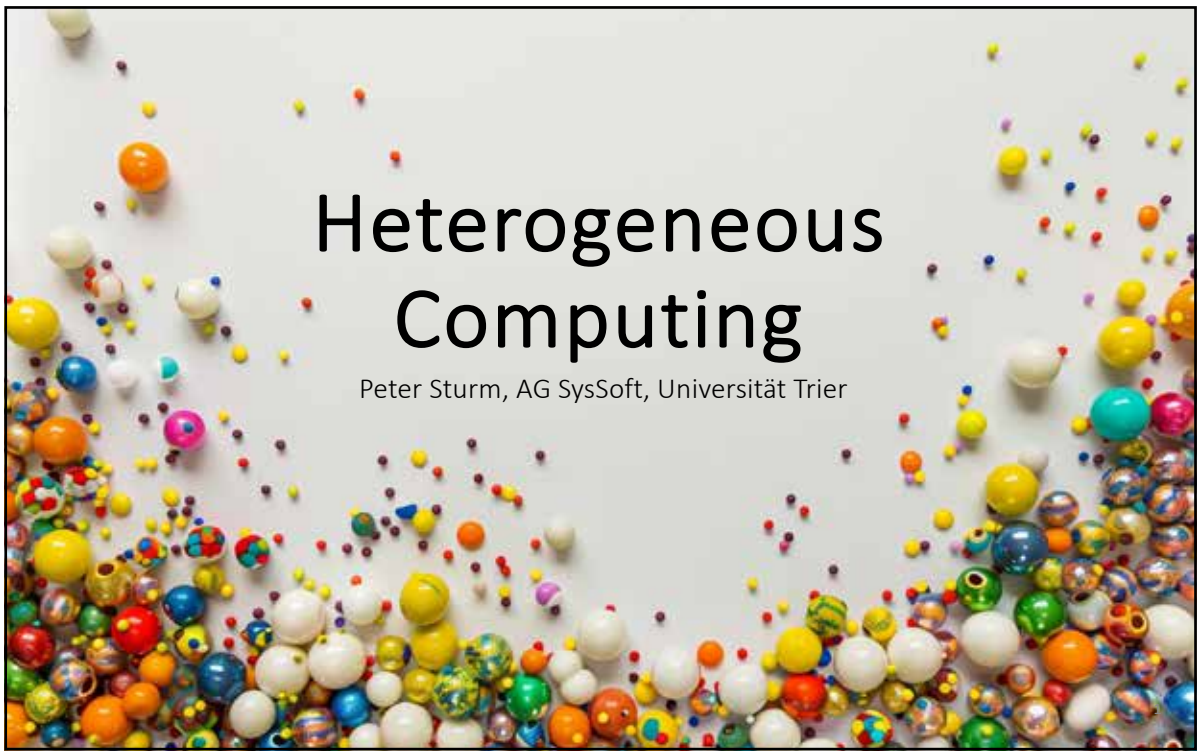

Heterogeneous
Computing
Peter Sturm, AG SysSoft, Universität Trier

4. ARM

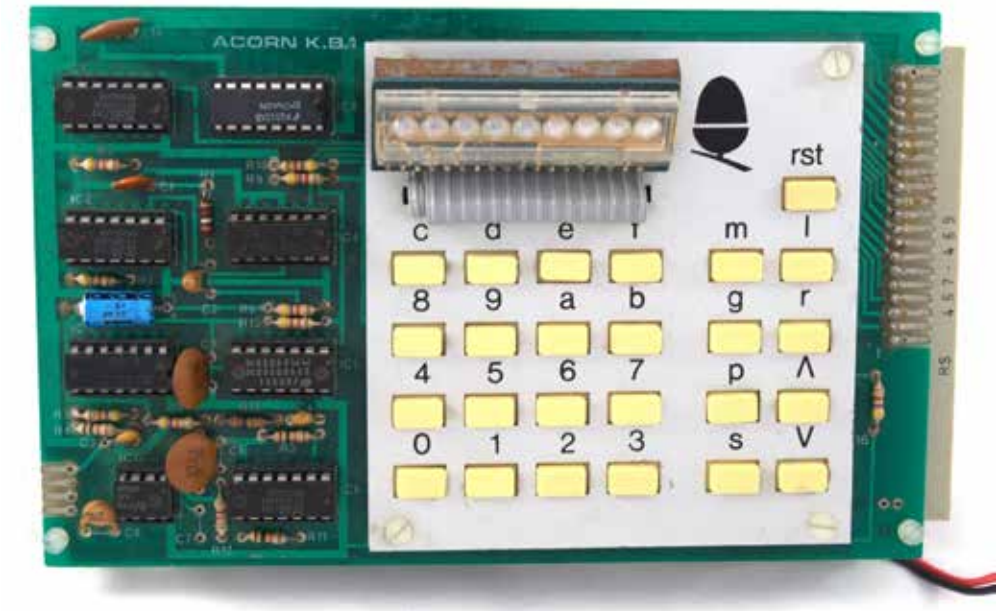
2

Historisches

- 1978 Acorn Computers Ltd., Cambridge UK
  - Hermann Hauser, Chris Curry
- 1979, Acorn System 1, CPU 6502
- 1980, Acorn Atom, CPU 6502
- 1982, BBC Micro, CPU 6502
  - Fernsehserie „The Computer Programme“

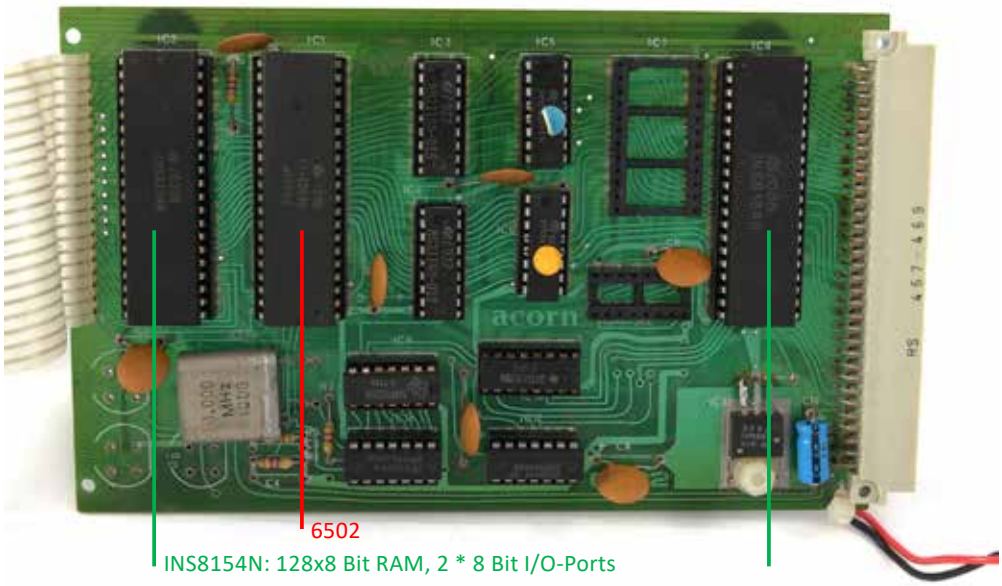
3

Acorn System 1 (Front)



4

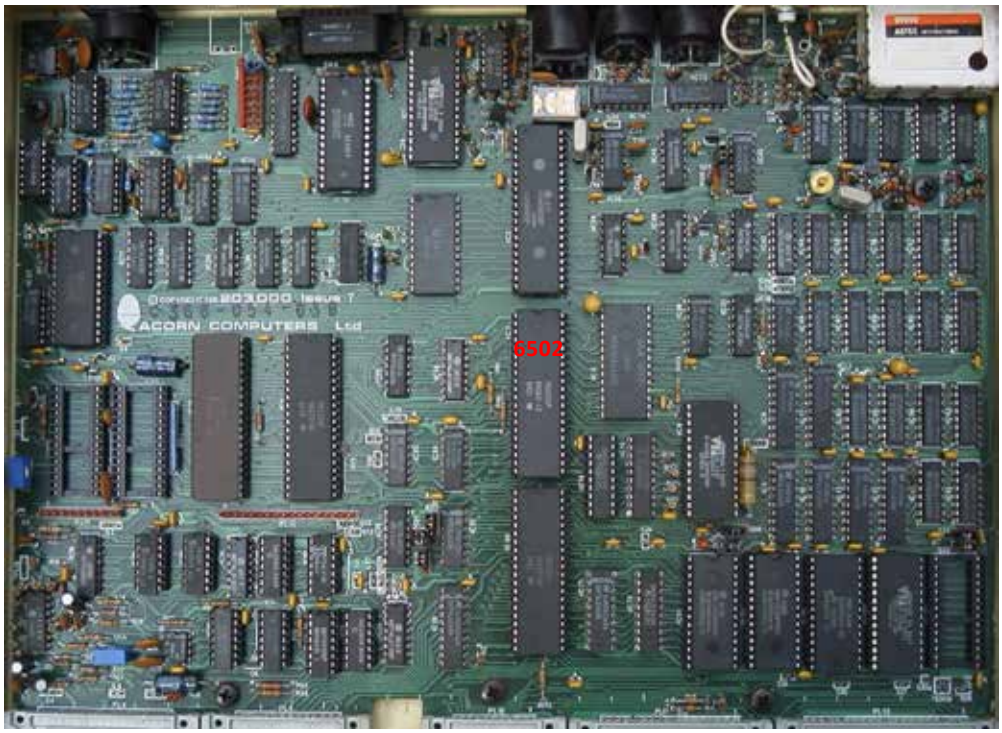
Acorn System 1 (Back)



5

BBC Micro

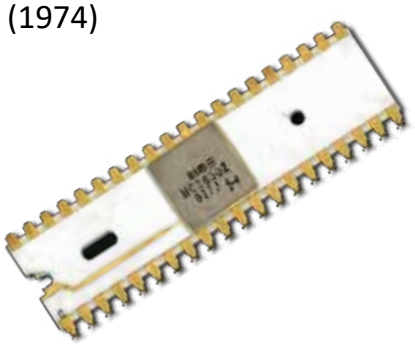
6



7

Seitenblick: 6502

- 1975, 8-Bit Prozessor von MOS Technology, $25
  - Intel 8080, ~$200 (1974)
  - Z-80 (1976)
- Angelehnt an Motorola 6800, ~$200 (1974)
- Sehr verbreitet
  - Acorn
  - Apple II
  - Comodore 64 und PET
  - Atari 8-Bit
  - Bender aus Futurama ☺



9

Apple II

Commodore PET

11

6502 als FPGA

Code:

	flops	slices	LUTs	RAM16	HDL	Notes
A2601	138	467	840	0	vhdl	by retromaster
Syntiac	144	564	1063	0	vhdl	by Peter Wendrich
RB6502	146	1005	1942	0	vhdl	by Ruud Baltissen (work in progress)
cpu.v	155	276	474	8	verilog	by Arlet Ottens
sprow	160	667	1224	0	vhdl	by Robert Sprowson (enhanced Free6502)
T65	162	547	985	0	vhdl	by Daniel Wallner et al
bc6502	179	544	951	0	verilog	by Rob Finch
6502_tc	293	1076	1995	0	vhdl	by Jens Gutschmidt
65c02_tc	317	1318	2460	0	vhdl	by Jens Gutschmidt
MyCPU	325	1612	2980	0	vhdl	by Dennis Kuschel, inspired by 6502

6502.org: (http://forum.6502.org/viewtopic.php?t=1673)

12

Historisches (2)

- 1982, 6502 wird langsam alt
- Suche nach Ersatz
  - Motorola 68000 war attraktiv, aber langsam bei Interrupts
  - Acorn ist mit keinem zufrieden
- 1983 eigener Befehlssatz entworfen
- 1985 ARM1, Acorn RISC Machine
- 1987 Acorn Archimedes, CPU ARM1

13

14

ARM Heute

15

... wenn es auch anders geht!

- Herausstellungsmerkmal: Geringer Stromverbrauch
- ARM verkauft Lizenzen
  - CPUs
  - GPUs (Mali)
- ARM Core License
  - AMD, Boradcom, Samsung, Huawei, IBM, Intel, TI, ...
- ARM Architectural License
  - Applied Micro, Broadcom, Huawei, Nvidia, AMD, Samsung, Apple, ...

17

Verkaufte Chips

Sales of chips containing Arm cores[131][132][133]

Year	Billion units	Relative size
2015	15	
2014	12	
2013	10	
2012	8.7	
2011	7.9	
2010	6.1	
2009	3.9	
2008	4.0	
2007	2.9	
2006	2.4	
2005	1.662	
2004	1.272	
2003	0.782	
2002	0.456	
2001	0.420	
2000	0.367	
1999	0.175	
1998	0.051	
1997	0.009	
Total	**78.094**	

Quelle: Wikipedia (https://en.wikipedia.org/wiki/Arm_Holdings)

18

Markets for ARM in 2017

	Devices Shipped (Million of Units)	2017 Devices	Device CAGR	Chips/ Device	2017 Chips	Chip CAGR	Key Growth Areas for ARM
Mobile	Smart Phone	1,700	20%	3-5	6,800	20%	←
	Feature Phone	-	-	-	-	-	
	Low End Voice	710	-1%	1-2	1,400	15%	
	Portable Media Players	90	-10%	1-3	180	-5%	
	Mobile Computing* (apps only)	850	20%	1	850	20%	←
Home	Digital Camera	130	-5%	1-2	200	-5%	
	Digital TV & Set-top-box	600	10%	1-4	2,000	25%	←
Enterprise	Desktop PCs & Servers (apps)	200	Flat	1	200	Flat	
	Networking	1,500	5%	1-2	1,700	5%	←
	Printers	130	2%	1-3	130	2%	
	Hard Disk & Solid State Drives	1,100	10%	1	1,100	10%	
Embedded	Automotive	3,800	10%	1	3,800	10%	
	Smart Card	8,500	10%	1	8,500	10%	
	Microcontrollers	11,400	5%	1	11,400	5%	←
	Others **	3,000	10%	1-2	3,000	10%	
	Total	**34,000**	**5%**		**41,000**	10%	

Source: Gartner, IDC, SIA, and ARM estimates

* Including tablets, netbooks and laptops ** Includes other applications not listed such as headsets, DVD, game consoles, etc

9 The Architecture for the Digital World® ARM

19

Naming

- Architekturen
  - ARMv{x}
  - Architecture Reference Manual
- Prozessoren
  - Klassisch
    - ARM{x}{labels}
    - ARM{x}{y}{z}{labels}
  - Seit 2004
    - ARM Cortex-{x}{y}
  - Technical Reference Manual

21

Architekturen & CPU-Familien

ARCHITECTURE	FAMILY
ARMv1	ARM1
ARMv2	ARM2, ARM3
ARMv3	ARM6, ARM7
ARMv4	StrongARM, ARM7TDMI, ARM8, ARM9TDMI
ARMv5	ARM7EJ, ARM9E, ARM10E, XScale
ARMv6	ARM11
ARMv6-M	Cortex-M0, Cortex-M0+, Cortex-M1
ARMv7-A	Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9, Cortex-A12, Cortex-A15
ARMv7-R	Cortex-R4, Cortex-R5, Cortex-R7
ARMv7-M	Cortex-M3
ARMv7E-M	Cortex-M4
ARMv8-A	Cortex-A53, Cortex-A57

22

{x}{y}{z}

X	Y	Z	DESCRIPTION	EXAMPLE
7			ARM7 core version	ARM7
9			ARM9 core version	ARM9
10			ARM10 core version	ARM10
11			ARM11 core version	ARM11
	1		Cache, write buffer and MMU	ARM710
	2		Cache, write buffer and MMU, Process ID support	ARM920
	3		Physically mapped cache and MMU	ARM1136
	4		Cache, write buffer and MPU	ARM940
	5		Cache, write buffer and MPU, error correcting memory	ARM1156
	6		No cache, write buffer	ARM966
	7		AXI bus, physically mapped cache and MMU	ARM1176
		0	Standard cache size	ARM920
		2	Reduced cache	ARM1022
		6	Tightly Coupled Memory	ARM1156
		8	As for ARM966	ARM968

23

ARM Labels

ATTRIBUTE	DESCRIPTION
D	Supports debugging via the JTAG interface. Automatic for ARMv5 and above.
E	Supports Enhanced DSP instructions. Automatic for ARMv6 and above.
F	Supports hardware floating point via the VFP coprocessor.
I	Supports hardware breakpoints and watchpoints. Automatic for ARMv5 and above.
J	Supports the Jazelle Java acceleration technology.
M	Supports long multiply instructions. Automatic for ARMv5 and above.
T	Supports Thumb instruction set. Automatic for ARMv5 and above.
-S	This processor uses a synthesizable hardware design.

24

ARM Cortex

- Cortex-A (Application processor)
  - General purpose CPU
  - Support für „richtige“ OS
  - Aktuelle Architektur: ARMv8-A / ARMv8.4-A
- Cortex-R (Real-time processor)
  - Minimal Interrupt-Latenz
  - Obere Schranken -> Echtzeitsysteme
  - Meist weniger leistungsfähig als Cortex-A
  - Aktuelle Architektur: ARMv7-R
- Cortex-M (Micro-controller)
  - Ultra-low-power, small form-factor
  - Aktuelle Architektur: ARMv7E-M

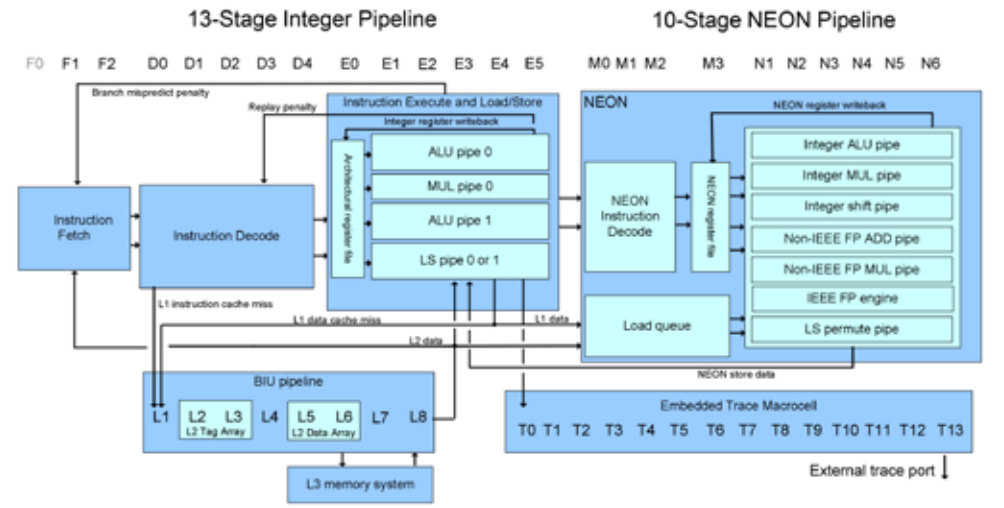
25

Varianten

- Hard Macro
  - PPA (Performance, Power, Area) optimiert
- Soft Macro
  - “Synthesizable“ (Sourcecode)
  - Mehr Freiheitsgrade
  - Höhere Flexibilität
  - PPA-Optimierung gegenüber Hard Macros meist schlechter

26

Full Cortex-A8 Pipeline Diagram



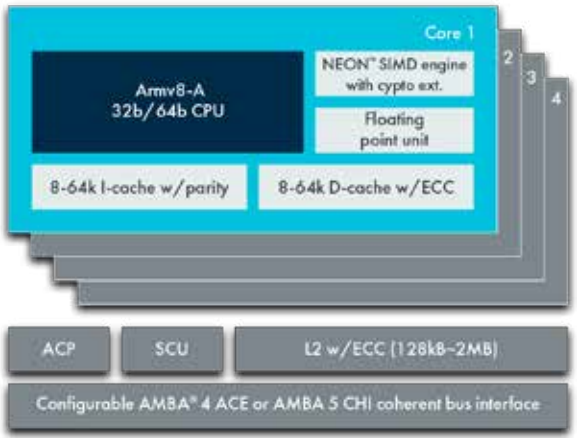
31 The Architecture for the Digital World® ARM

27

Raspberry Pi 3 B+

28

CPU BCM2837B0



- Cortex-A53 (ARMv8), 64 Bit, 1.4GHz

29

ARM Assembler

30

Register

- 16 General Purpose Register (r0 ... r15)
- Banking
  - r0 ... r7 immer sichtbar
  - FIQ (Fast IQ): r8 ... R14 überlagert
  - Supervisor, IRQ: r13 und r14
- r13 Stack Pointer
- r14 Link Register
  - Rücksprungadresse
- r15 PC

31

AAPCS

- ARM Architecture: Procedure Call Standard
- Argumente
  - r0 ... r3, danach Stack
- Inhalte von r4 ... r11 „bleiben“ unverändert
  - Unterprogramm muß ggf. auf Stack sichern
- r12 Scratch Register (darf jeder ändern)
- Rückgabe
  - r0 (ggf. r1)

32

CPSR

- Current Program Status Register
- Current processor mode
- Interrupt disable flags
- Current processor state (ARM, Thumb, ...)
- Data memory endianness (for ARMv6 and later)
- Condition flags
  - N, Z, C, V (Overflow)

33

Thumb / Thumb-2

- ARM ISA
  - Volle RISC Befehlssatz
- Thumb
  - 16 Bit Format für eine Teilmenge der ARM ISA
  - Einschränkungen
    - Conditional nur bei Branches
- Thumb-2
  - Erweitert Thumb um 32 Bit Instruktionen
  - Deckt kompletten Befehlssatz ab

34

„Nun ja“

- Cortex-M versteht nur Thumb/Thumb-2

M0	M0+	M1	M3	M4	SIZE	INSTRUCTIONS
Yes	Yes	Yes	Yes	Yes	16	ADC, ADD, ADR, AND, ASR, B, BIC, BKPT, BLX, BX, CMN, CMP, CPS, EOR, LDM, LDR, LDRB, LDRH, LDRSB, LDRSH, LSL, LSR, MOV, MUL, MVN, NOP, ORR, POP, PUSH, REV, REV16, REVSH, ROR, RSB, SBC, SEV, STM, STR, STRB, STRH, SUB, SVC, SXTB, SXTH, TST, UXTB, UXTH, WFE, WFI, YIELD
Yes	Yes	Yes	Yes	Yes	32	BL, DMB, DSB, ISB, MRS, MSR
No	No	No	Yes	Yes	16	CBNZ, CBZ, IT
No	No	No	Yes	Yes	32	ADC, ADD, ADR, AND, ASR, B, BFC, BFI, BIC, CDP, CLREX, CLZ, CMN, CMP, DBG, EOR, LDC, LDMA, LDMDB, LDR, LDRB, LDRBT, LDRD, LDREX, LDREXB, LDREXH, LDRH, LDRHT, LDRSB, LDRSBT, LDRSHT, LDRSH, LDRT, MCR, LSL, LSR, MLS, MCRR, MLA, MOV, MOVT, MRC, MRRC, MUL, MVN, NOP, ORN, ORR, PLD, PLDW, PLI, POP, PUSH, RBIT, REV, REV16, REVSH, ROR, RRX, RSB, SBC, SBFX, SDIV, SEV, SMLAL, SMULL, SSAT, STC, STMIA, STMDB, STR, STRB, STRBT, STRD, STREX, STREXB, STREXH, STRH, STRHT, STRT, SUB, SXTB, SXTH, TBB, TBH, TEQ, TST, UBFX, UDIV, UMLAL, UMULL, USAT, UXTB, UXTH, WFE, WFI, YIELD
No	No	No	No	Yes	32	PKH, QADD, QADD16, QADD8, QASX, QDADD, QDSUB, QSAX, QSUB, QSUB16, QSUB8, SADD16, SADD8, SASX, SEL, SHADD16, SHADD8, SHASX, SHSAX, SHSUB16, SHSUB8, SMLABB, SMLABT, SMLATB, SMLATT, SMLAD, SMLALBB, SMLALBT, SMLALTB, SMLALTT, SMLALD, SMLAWB, SMLAWT, SMLSD, SMLSLD, SMMLA, SMMLS, SMMUL, SMUAD, SMULBB, SMULBT, SMULTT, SMULTB, SMULWT, SMULWB, SMUSD, SSAT16, SSAX, SSUB16, SSUB8, SXTAB, SXTAB16, SXTAH, SXTB16, UADD16, UADD8, UASX, UHADD16, UHADD8, UHASX, UHSAX, UHSUB16, UHSUB8, UMAAL, UQADD16, UQADD8, UQASX, UQSAX, UQSUB16, UQSUB8, USAD8, USADA8, USAT16, USAX, USUB16, USUB8, UXTAB, UXTAB16, UXTAH, UXTB16

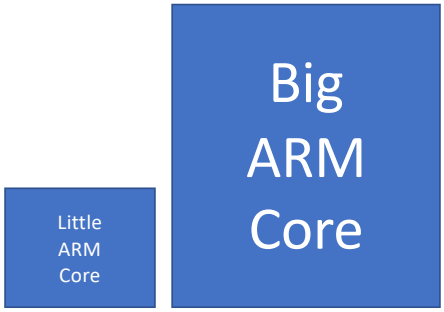
35

Thumb-2 / IF ... THEN

```
CMP r0, r1 ; r0 == r1?
ITEET EQ ; was the result EQ? Then / Else / Else / Then
MOVEQ r0, r4 ; If r0 == r1, execute this instruction
MOVNE r0, r5 ; Else execute this instruction
SUBNE r0, #1 ; Else execute this instruction too
BEQ label ; If r0 == r1, branch
```

36

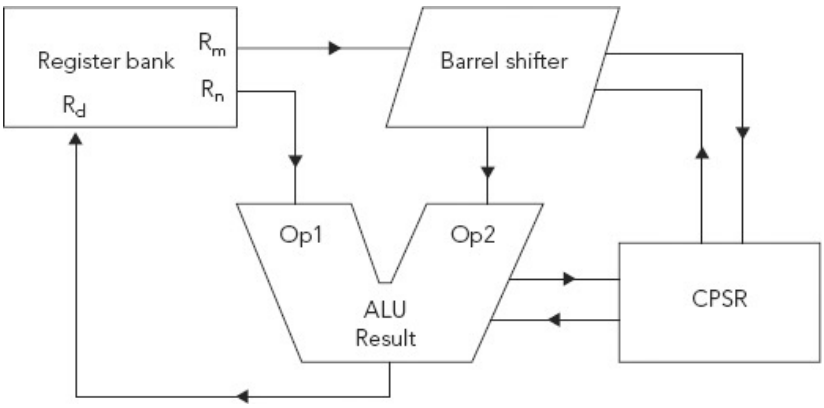
Big.LITTLE



- Minimaler Energieverbrauch
- Zwei unterschiedliche schnelle aber identische Prozessoren
  - Kontext entscheidet, welche CPU rechnet

37

Architekturkern



aus „Professional Embedded ARM Development“, James A. Langbridge, Wrox, 2014

38

Instruktionsformat

```
<op>{cond}{flags} Rd, Rn, Operand2
```

- op = 3 Zeichen Mnemonic
- cond = Bedingte Ausführung
- flags
  - S: Update CPSR-Flags
- Rd = Zielregister
- Rn = Erstes Register
- Operand2 = Zweites Register oder zweiter Operand

39

Conditions

- AL – Always (default)
- NV – Never
- EQ - Equal
- NE – Not Equal
- VS – Overflow Set
- VC – Overflow Clear
- MI – Minus
- PL – Plus
- CS – Carry Set
- CC – Carry Clear
- HI – Higher
- LS – Lower or Same
- GE – Greater Than or Equal
- LT – Less Than
- GT – Greater Than
- LE – Less Than or Equal

40

Adressierungsarten

- Immediate: #2a
- Register: r7
- Register Indirekt: [r3] (nicht für Rd)
  - mit immediate Displacement: [r3 #2]
  - mit Offset in Register: [r3 r4]
  - Default post-index (pre-index: !)

41

Beispielprogramm

```
sum
      MOV r1,#0
sum_loop
      ADD r1,r1,r0
      SUBS r0,r0,#1
      BNE sum_loop
sum_rtn
      MOV r0,r1
      MOV pc,lr
```

42

Literatur

6666 Seiten

43

Zu finden auf: http://infocenter.arm.com/help/index.jsp

Raspberry Pi 3 B+

635 Seiten

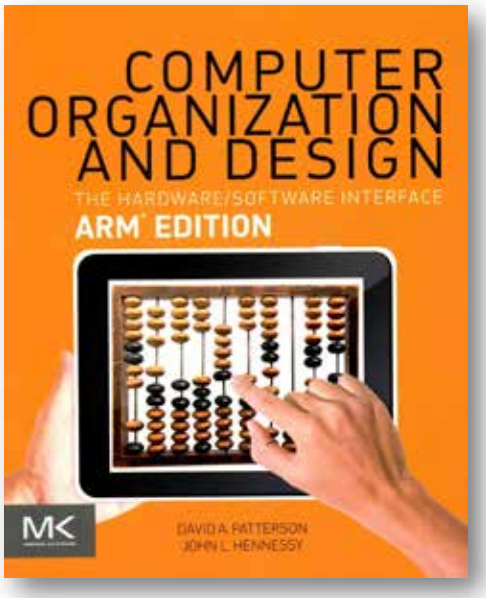
44

Arduino Due

1459 Seiten

45

Auch in der Lehre?



46