Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

Vorlesung: Informationstechnik (IT)

Institutsleitung
Prof. Dr.-Ing. K. D. Müller-Glaser
Prof. Dr.-Ing. J. Becker
Prof. Dr. rer. nat. W. Stork

Prof. Dr.-Ing. K.D. Müller-Glaser

Institut für Technik der Informationsverarbeitung (ITIV)

Rechnerarchitektur (II)

KIT – Universität des Landes Baden-Württemberg und
nationales Forschungszentrum in der Helmholtz-Gemeinschaft

www.kit.edu

Inhalt

- Mikroprozessoren
- Mikrocontroller
- Digitale Signalprozessoren
- Anwendungsspezifische Prozessoren

- Bespiele

- Atmel
- ARM
- Intel

Andauernd
Forderung nach Leistungssteigerung

- Höhere Taktfrequenzen
- Pipelining
- Mehr Speicher, größere Speicherworte
- Multi Level Caches
- Superskalar
- Multi-Prozessor / Multi-Core Systeme
  - SISD, SIMD
  - MIMD, MISD

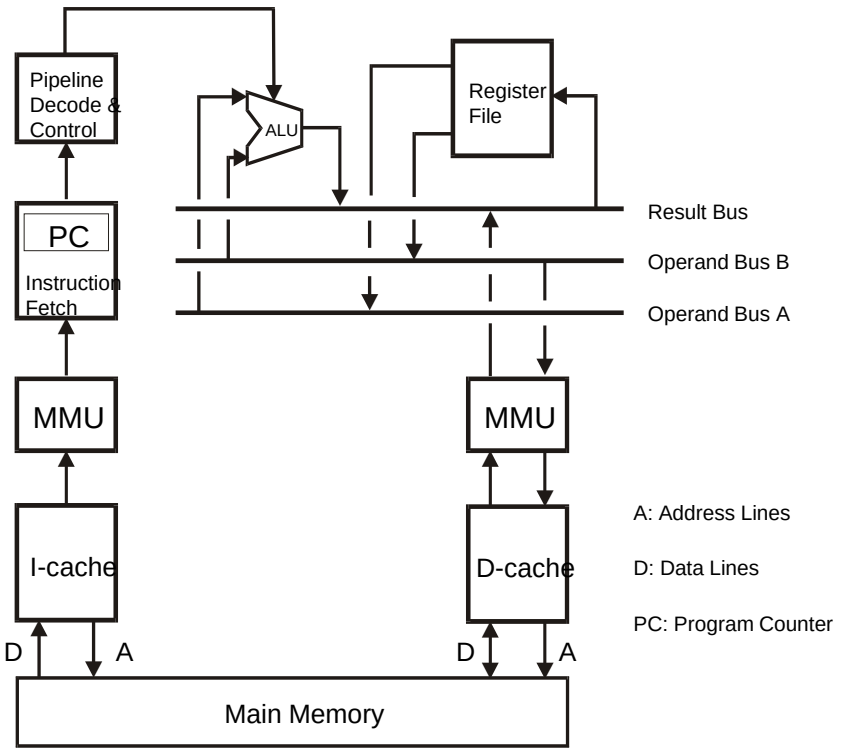
Optimierung Rechenleistung

- Rechenleistung
- Energieverbrauch (Power Dissipation)
- Flächenbedarf (Chipkosten)
- General Purpose versus
  Application Specific Standard Prozessoren
- Mikroprozessoren
- Mikrocontroller
- Digitale Signalprozessoren

Entwicklung Hochleistungsprozessoren

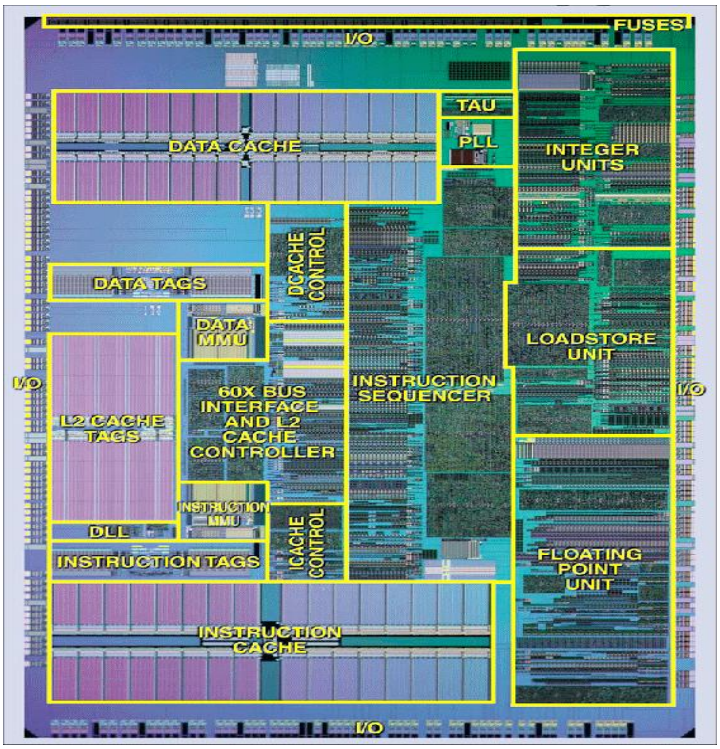
	CISC		RISC				EPIC	VLIW	
		Intel							
		…							
1978	**Motorola**	8086							
1979	6800						siehe	Mosberger	
1980					Berkeley	Stanford			
1981					RISC I/II	MIPS			
1982									
1983									
1984									
1985		80386	**HP**		**SUN**				
1986	68020		PA-RISC		**FUJITSU**				
1987					SPARC			Cydrome	Multiflow
1988				**IBM**				Cydra5	TRACE
1989				POWER		**DEC**			
1990						**Compaq**			
1991						Alpha			
1992				POWER2					
1993		Pentium							
1994	68060								
1995		Ppro			Ultra	21164			
1996			PA8000		SPARC				
1997		Pentium II	PA-8200		Ultra				
1998			PA-8500	POWER3	SPARC II	21264			
1999		Pentium III					**Intel & HP**		
2000			PA-8600		Ultra		IA64		
2001		Pentuim IV	PA8700	POWER4	SPARC III		Itanium		
2002									
2003		Pentium M	PA8800				Itanium 2		
2004					SPARC64 V	EV7z			
2005		Pentium D	PA-8900				Dual-Core		
2006		Core 2 Duo					Itanium 2		
2007		Core 2 Quad		POWER6	SPARC64 VI				
2008		Core i7			SPARC64 VII				
2009									

Struktur eines RISC Prozessors

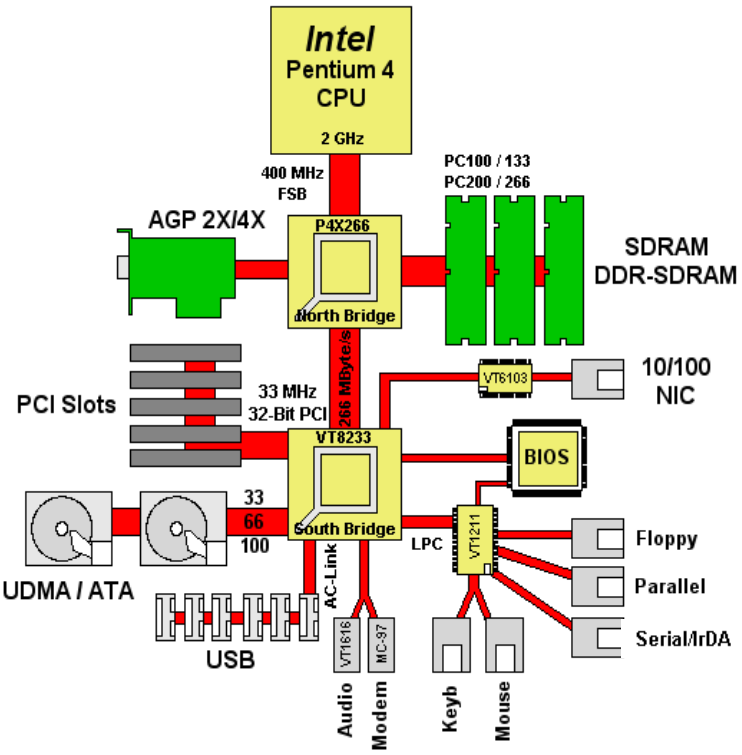


Microelectronics: today's processors

IBM Power PC 750



PC Mainboard Architecture (Stand 2004)



PCI Peripheral Component Interconnect

AGP Accelerated Graphics Port

NIC Network Information Center

IrDA Infrared Data Assocoiation

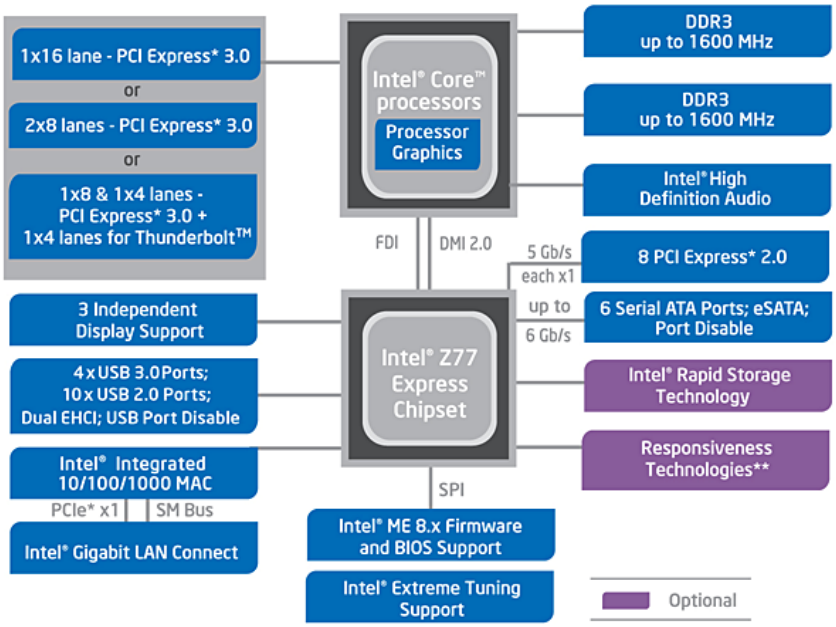
UDMA Ultra DMA Direct Memory Access

ATA Advanced Technology Attachement

USB Universal Serial Bus

EIDE Enhanced Integrated Disc Electronics

PC Mainboard Architecture (Stand 2013)



* Other Names and Brands maybe claimed as property of others.

**Includes Intel® Smart Response Technology, Intel® Rapid Start Technology, and Intel® Smart Connect Technology

PCI Peripheral Component Interconnect

AGP Accelerated Graphics Port

NIC Network Information Center

IrDA Infrared Data Assocoiation

UDMA Ultra DMA Direct Memory Access

ATA Advanced Technology Attachement

USB Universal Serial Bus

EIDE Enhanced Integrated Disc Electronics

Prozessoren für Desktop und Serversysteme

- Anforderungen an Mikroprozessoren :
  - Geeignet für allgemeine Datenverarbeitung (General Purpose µP) PC, Server, Rechenzentrum
  - Höchste Rechenleistung, großer Hauptspeicher, höchste Datentransferraten
  - Hohe Preise werden akzeptiert
  - Hoher Energiebedarf wird in Kauf genommen, Problem Kühlung
  - Skalierbarkeit (gleicher Kern mit verschiedener Ausstattung)
  - Benötigt Chipsatz für Aufbau eines Rechners
  - hohe Anforderungen an Kompatibilität
- Marktführer Intel Corporation, AMD

Prozessoren für eingebettete Systeme

- Anforderungen an Prozessoren sind anders als bei Desktop-/Server-Systemen:
  - Aufgabe: Messen Steuern Regeln, weniger allg. Datenverarbeitung
  - Geringe Systemkosten
  - Integrierte I/O-Komponenten ("System on a Chip“, Mikrocontroller)
  - Niedriger Stromverbrauch wichtiger als Performance
  - Geringer Energiebedarf, geringe Abwärme
  - Skalierbarkeit (gleicher Kern mit verschiedener Ausstattung)
  - Geringere Anforderungen an Kompatibilität
- Gewaltiger Markt: Weit über 90% der Jahresproduktion an Prozessoren
- Breites Spektrum an Technologie und Leistungsfähigkeit
  - Von 4 Bit bis zu mehr als 64 Bit
  - Von weniger als einem KB adressierbaren Speicher bis hin zu vielen TB

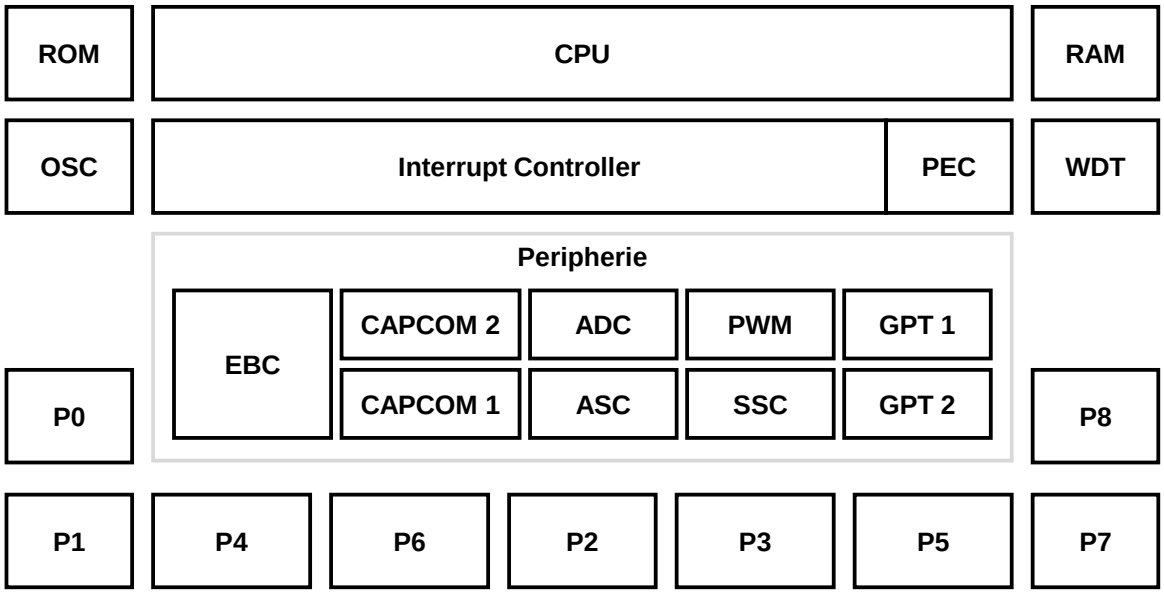
Prozessoren für eingebettete Systeme

- Sehr große Vielfalt von Herstellern und Typen (kleine Auswahl):
  - Motorola: 68k-Architektur (CPU32)
  - Motorola und IBM: PowerPC
  - ARM: Verschiedene Versionen der ARM-Architektur und diverse Lizenznehmer
  - Hitachi: SH3, SH4 und SH5
  - NEC und andere: CPUs auf Basis der MIPS-Architektur
  - Microchip: PIC
  - Intel: i860, i960, ATOM und diverse andere (u.a. ARM)
  - Texas Instruments: MSP43O und andere
  - Infineon: C16x und andere
  - Atmel: AVR
  - Viele weitere Architekturen und Lizenzproduktionen

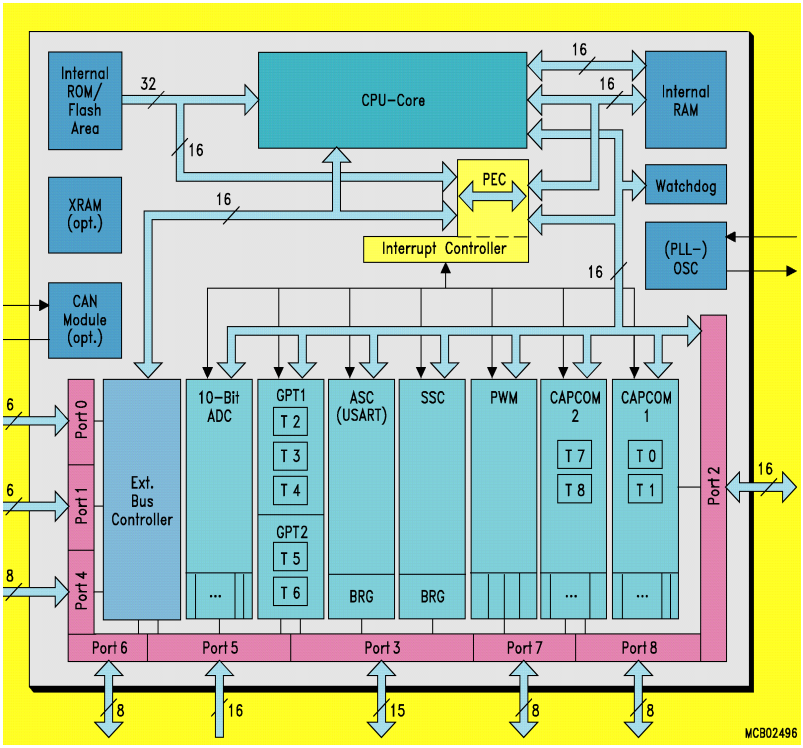
Mikrocontroller

Infineon C167

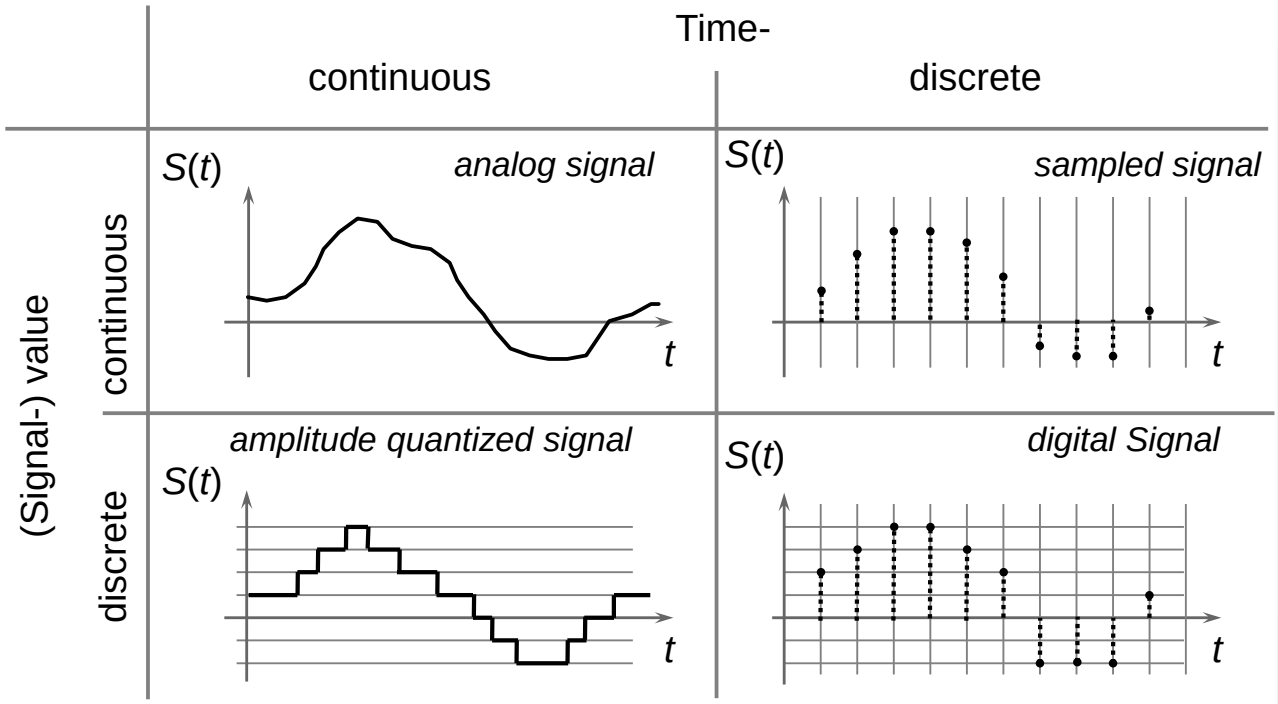
Blockschaltbild:



Block Diagram Infineon C167 microcontroller



Signal classes



Nachrichtentechnische Kanäle lassen sich in die obigen Signalklassen einordnen. Die Signalklassen machen eine Aussage über den Signalverlauf, welcher durch seinen Signalwert-Verlauf (y-Achse *s*) über die Zeit (x-Achse *t*) bestimmt ist. Die unterschiedlichen Signalklassen ergeben sich aus der Kombination des Wert- und Zeitverlaufs:

- Kontinuierlich: stetiger Verlauf (kein Abstand zwischen je zwei Punkten)
- Diskret: sprunghafter Verlauf (Einschränkung auf bestimmte Werte)

Nun kann der kontinuierliche und der diskrete Fall auf den Signalwertverlauf und auf den Zeitverlauf sinnvoll angewendet werden und in beliebigen Kombinationen auftreten. Insgesamt können dabei
2·2 = 4 Signalklassen unterschieden werden:

Signal- und zeitkontinuierlich: z.B. analoges Telefon, Rundfunk: Weder der Signalwert, noch der Zeitverlauf wird zerhackt. Das entspricht dem klassischen analogen Signal

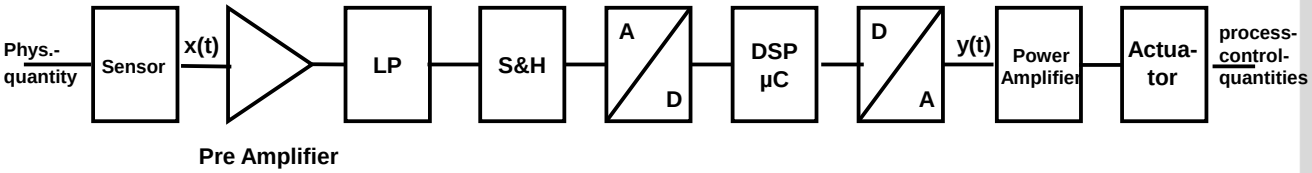
- Signalkontinuierlich und zeitdiskret: z.B. periodisches Messen von analogen Werten eines technischen Prozesses: Der technische Prozess könnte z.B. ein Überwachungsprozess sein, in dem von einem Sensor alle 10 Sekunden oder auch auf Anfrage die bestehende Lichtintensität in [Lux] gemeldet wird.
- Signaldiskret: digitale Übertragung mit beliebigen Signalwechseln (zeitkontinuierlich) oder festem (meist isochronem) Taktraster (zeitdiskret). In diesem Fall spricht man von einem digitalen Signal

Die zweiwertige digitale Übertragung (Binärübertragung) ist ein Spezialfall des signaldiskreten Verlaufs, da hier die zulässigen Signalwerte auf zwei begrenzt werden.

Es gibt mehrere Verfahren im Bereich der Telekommunikation, um ein Signal einer bestimmten Klasse in ein Signal einer anderen Klasse zu wandeln. Hierbei ist die Wandlung vom kontinuierlichen zum diskreten Fall von besonderer Relevanz: Die Digitalisierung in der Telekommunikation bedeutet, dass signal-/zeitkontinuierliche Signalklassen in entsprechende diskrete Signalklassen gewandelt werden. Das bedeutendste und am meisten verbreitete Telekommunikationssystem, welches eine solche Wandlung benutzt, ist das Fernsprechen, also das Telefonnetz. Zur Analog-Digitalwandlung wird dort das Pulse Code

Digital signal processing in mechatronic systems

- Basic structure of digital signal processing systems



Typical Algorithms:

- ❒ Digital filters (FIR, IIR)
- ❒ Transformations (Windowing, DFT, FFT)
- ❒ Correlation

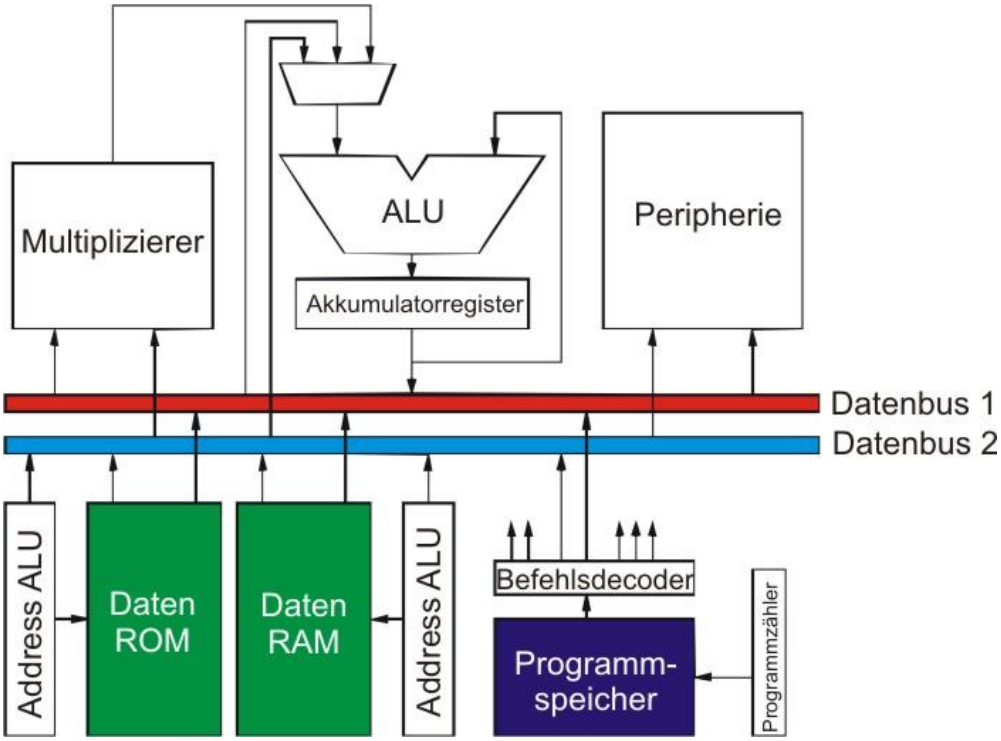
LP: Low Pass
S&H: Sample and Hold
ADC: Analog/Digital Converter
DSP: Digital Signal Processor
µC: Microcomputer
DAC: Digital/Analog Converter
FIR: Finite Impulse Response
IIR: Infinite Impulse Response
DFT: Digital Fourier Transform
FFT: Fast Fourier Transform

Digital Signal Processing

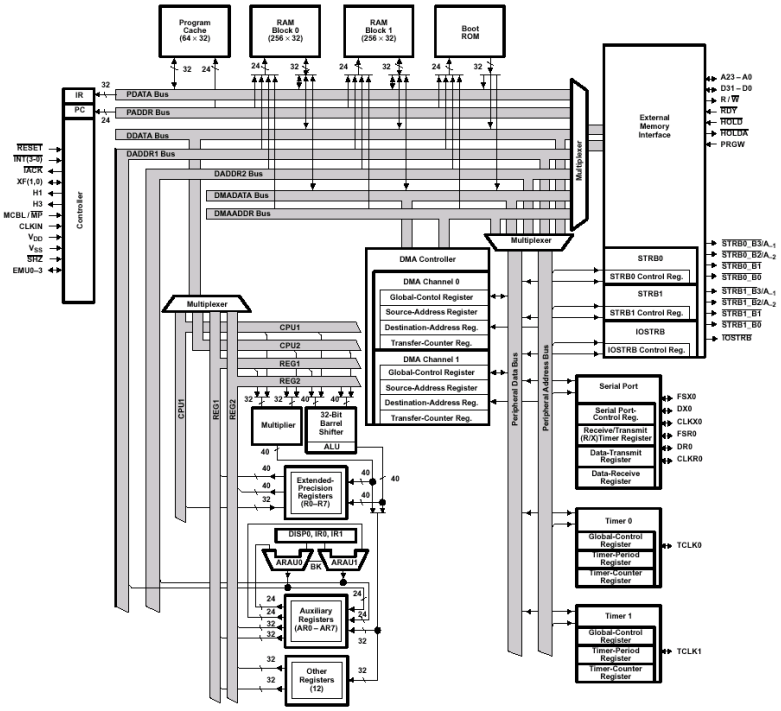
- Key operation for most algorithms: Sum of Products

Algorithm	Equation
Finite Impulse Response Filter	$y(n)=\sum_{k=0}^{M} a_k\, x(n-k)$
Infinite Impulse Response Filter	$y(n)=\sum_{k=0}^{M} a_k\, x(n-k)+\sum_{k=1}^{N} b_k\, y(n-k)$
Convolution	$y(n)=\sum_{k=0}^{N} x(k)h(n-k)$
Discrete Fourier Transform	$X(k)=\sum_{n=0}^{N-1} x(n)\exp[-j(2\pi/N)nk]$
Discrete Cosine Transform	$F(u)=\sum_{x=0}^{N-1} c(u).f(x).\cos\left[\frac{\pi}{2N}u(2x+1)\right]$

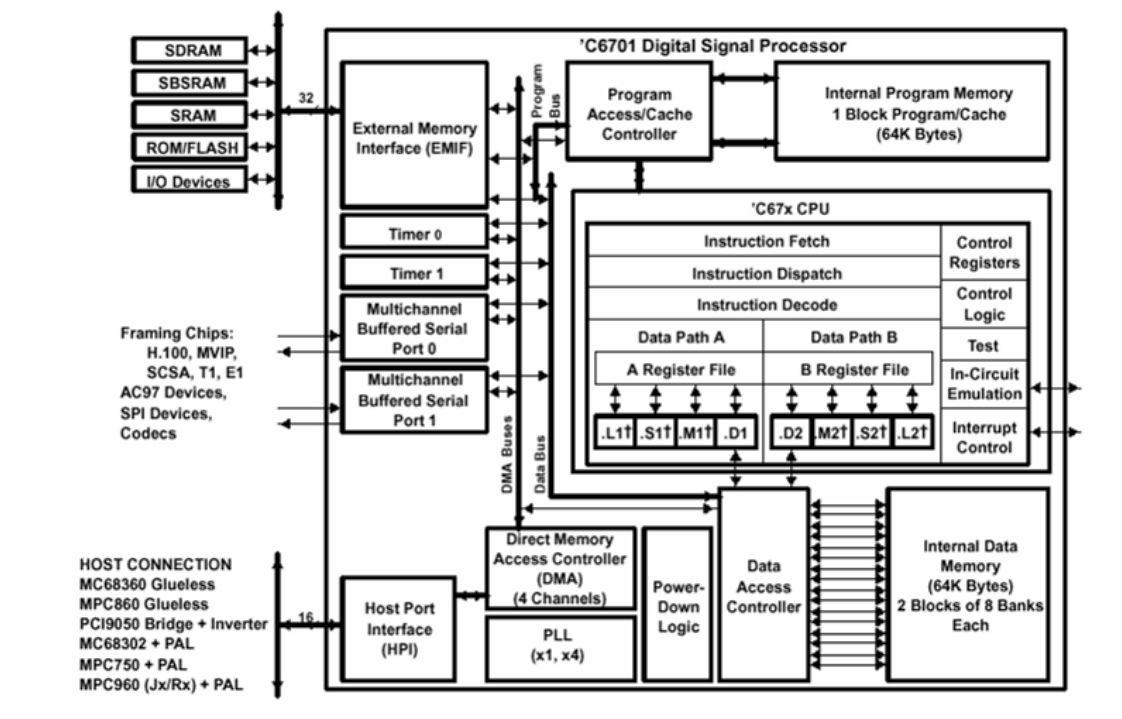
Vereinfachtes Blockschaltbild eines DSP



Block Diagram Texas Instruments TMS320C32

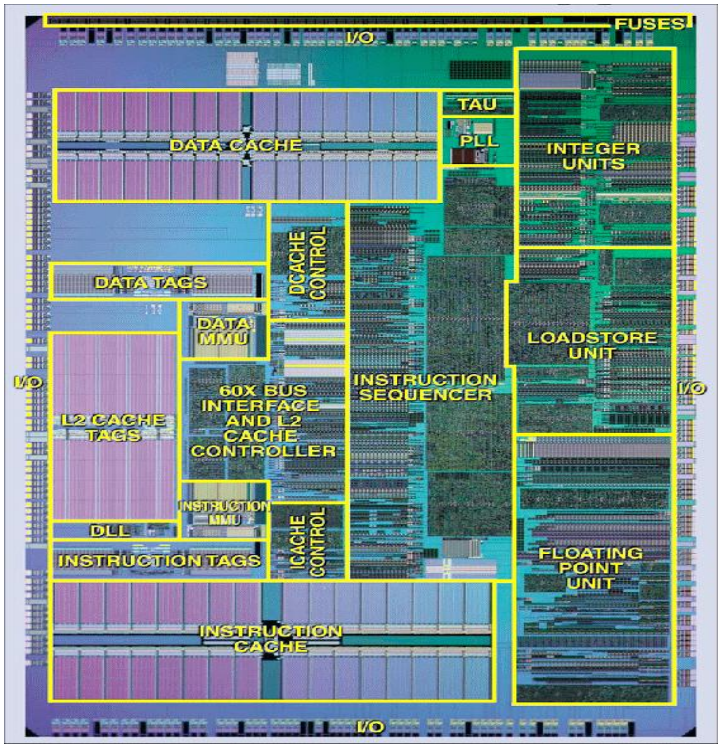


Architecture TMS320C6701



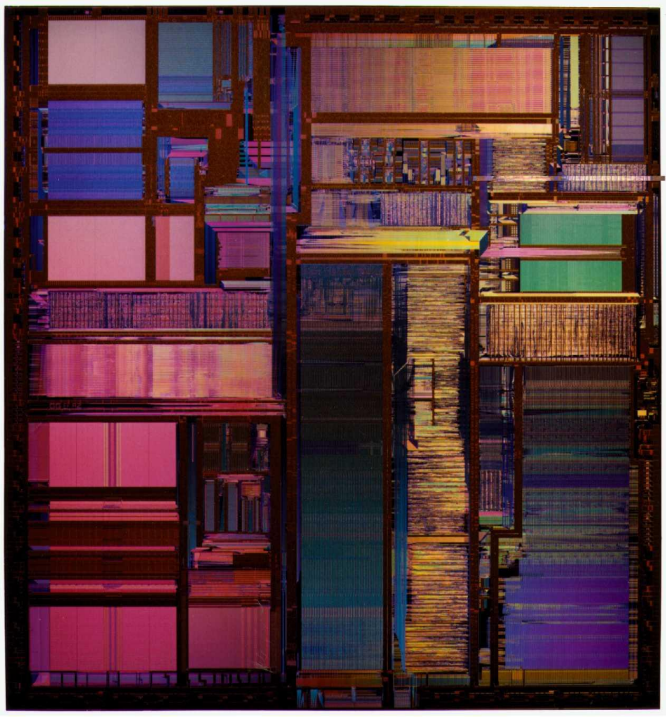
Microelectronics: today's processors

IBM Power PC 750



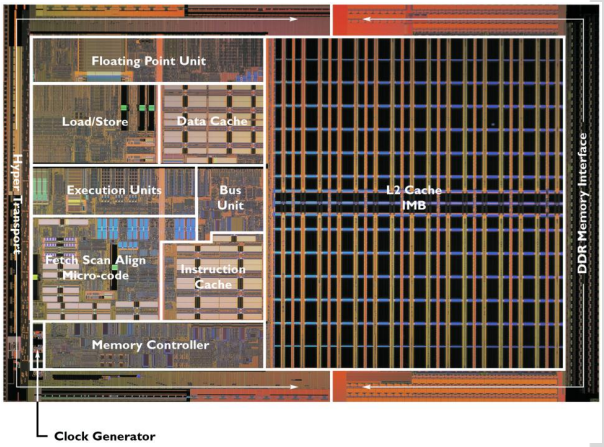
Microelectronics: today's processors

Intel Pentium Processor



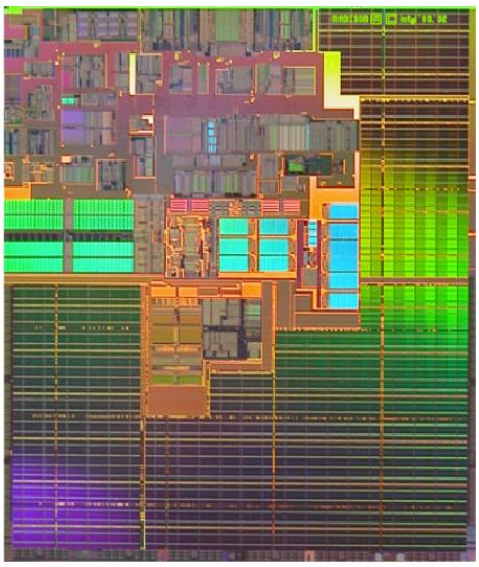
Rechnerarchitekturen

- Beispiel für eine CISC Maschine: AMD Opteron Prozessor
  - Taktfrequenz 1,5 – 2,2 GHz
  - AMD64 Instruktionen, MMX, 3DNow, SSE, SSE2 Support
  - 64 KByte Level 1 Cache, 2-Wege assoziativ
  - 1 MByte Level 2 Cache, 16-Wege assoziativ
  - 48-bit virtueller Adressraum, 40-bit physikalisch
  - HyperTransport Anbindung zwischen CPUs und Systemkomponenten



Intel High-End Processor: Itanium

130nm Itanium® 2 Processor Highlights

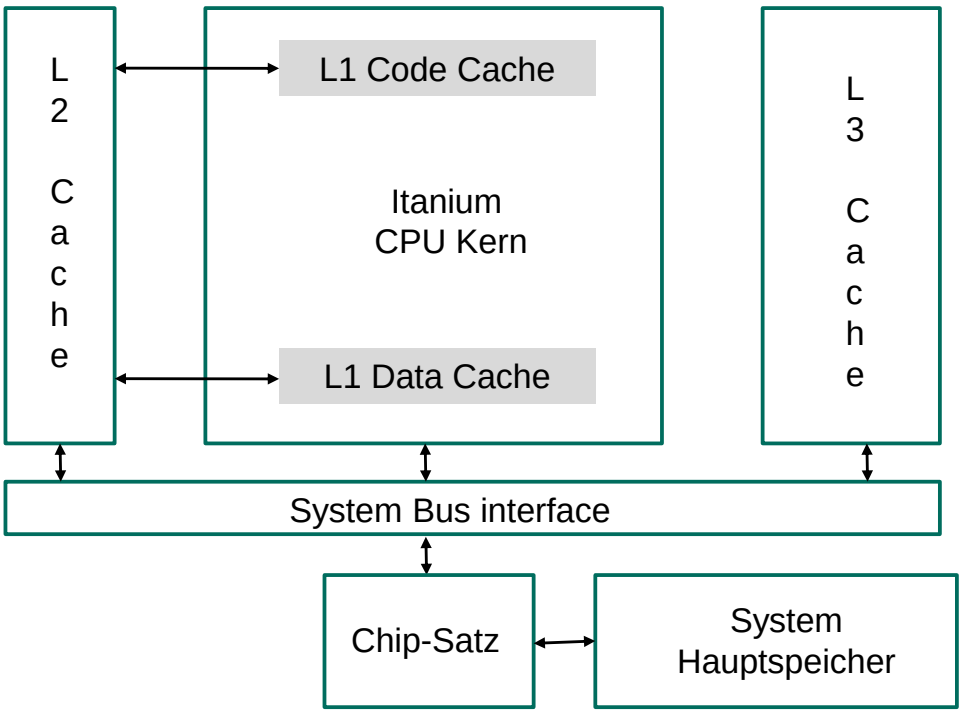


- 410M transistors
- 374mm² die size
- 6MB on-die L3 cache
- 1.5GHz at 1.3V
- 6.4GB/s 400MT/s 4-way bus interface
- Plug-in compatible with existing platforms
- Extensive RAS, DFT and DFM features

Largest microprocessor transistor count and on-die cache

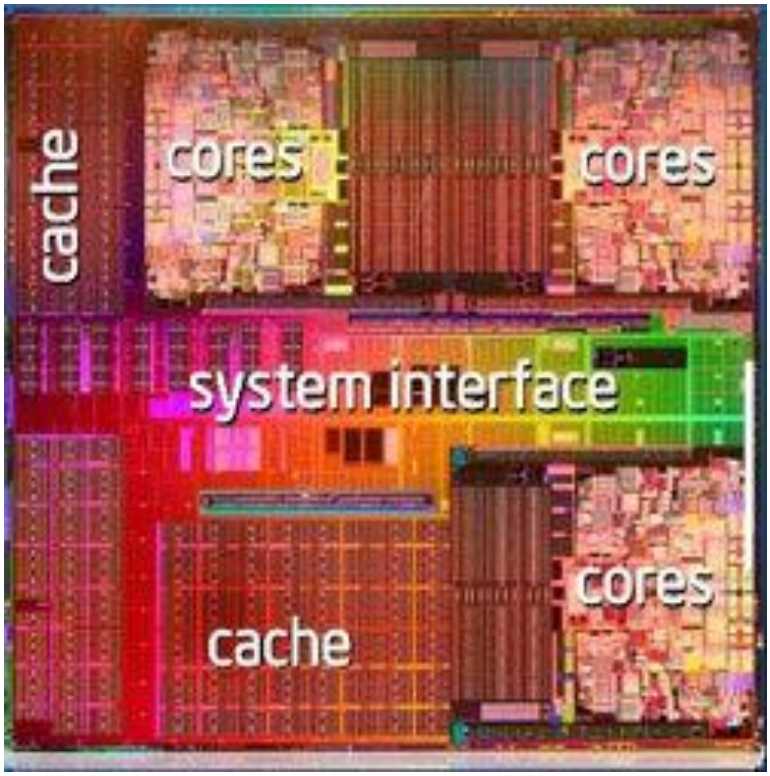
2003

Intel IA64-Prozessoren Cache Organisation

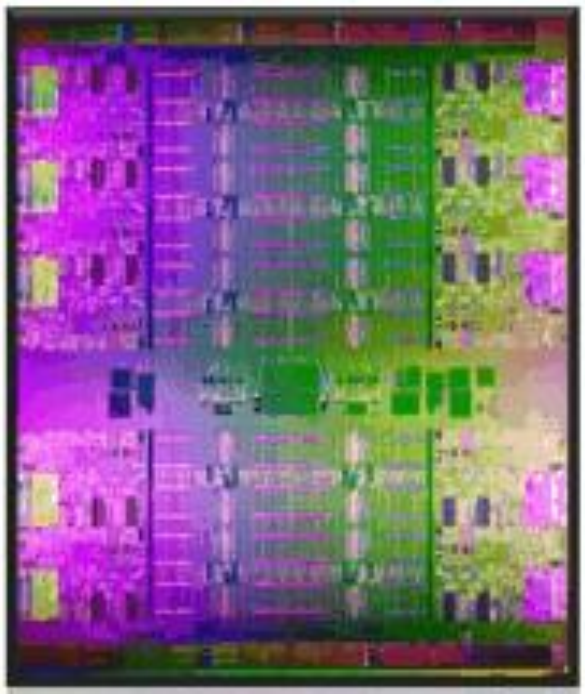


Dunnington:

- Auf einem Chip 6 Penryn-Kerne
- 32 KB L1-Cache pro Kern
- 3 MByte L2-Cache pro Doppelprozessor
- 16 MByte L3-Cache
- 1,8 Milliarden Transistoren **2009**

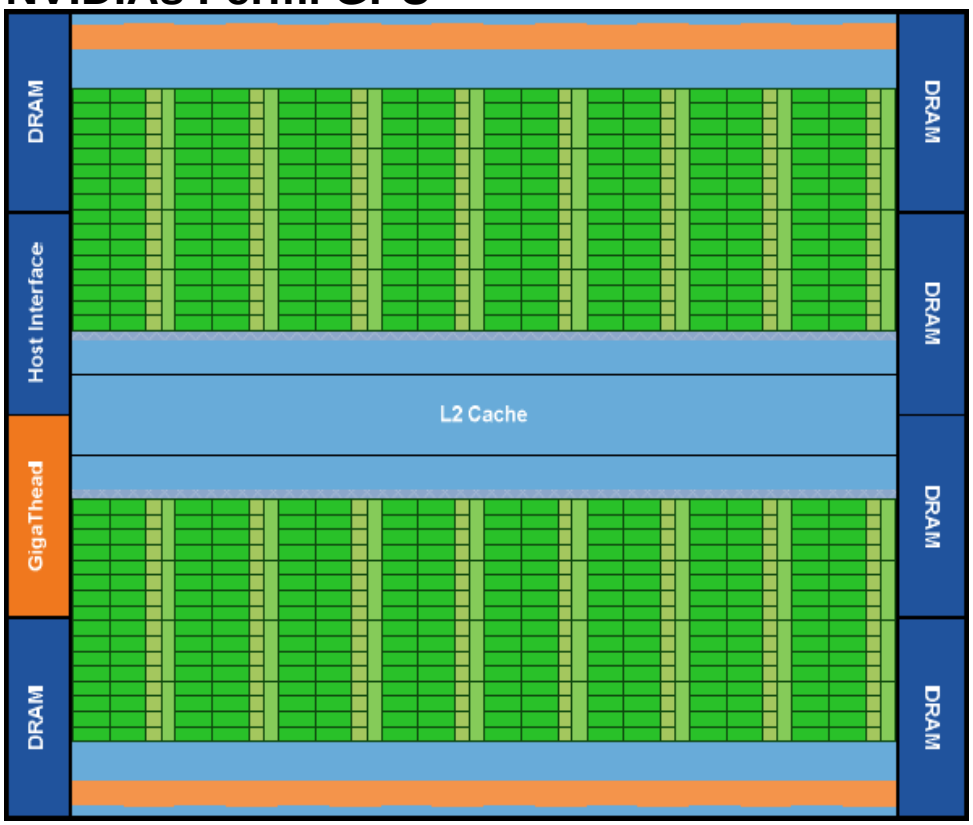


Intel Prozessoren



- Westmere-EX
- 10 Kerne, 20 logische Kerne
- 30 MByte L3-Cache
- 2,9 Milliarden Transistoren
- 32 nm Strukturen
- Frühjahr **2011**

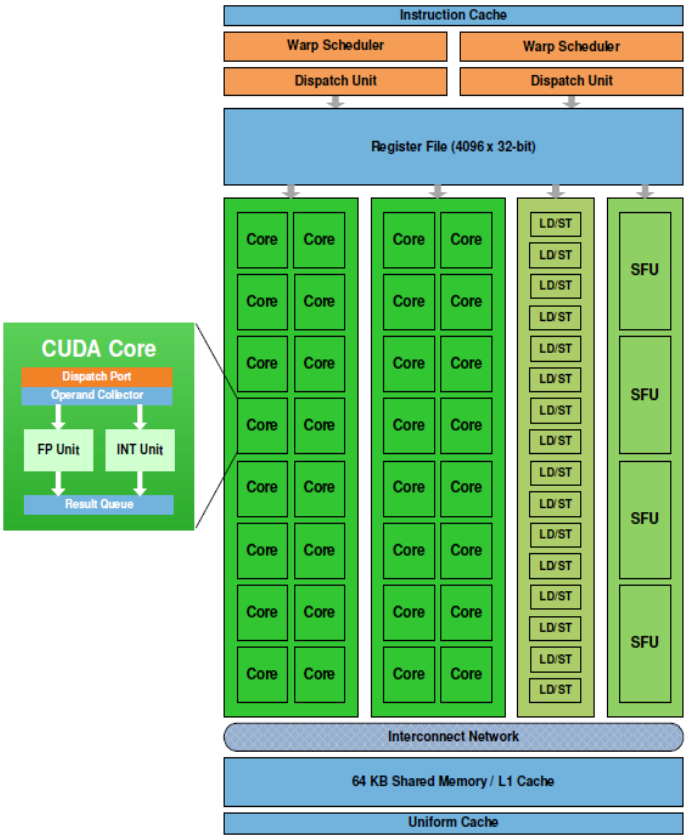
NVIDIAs Fermi GPU



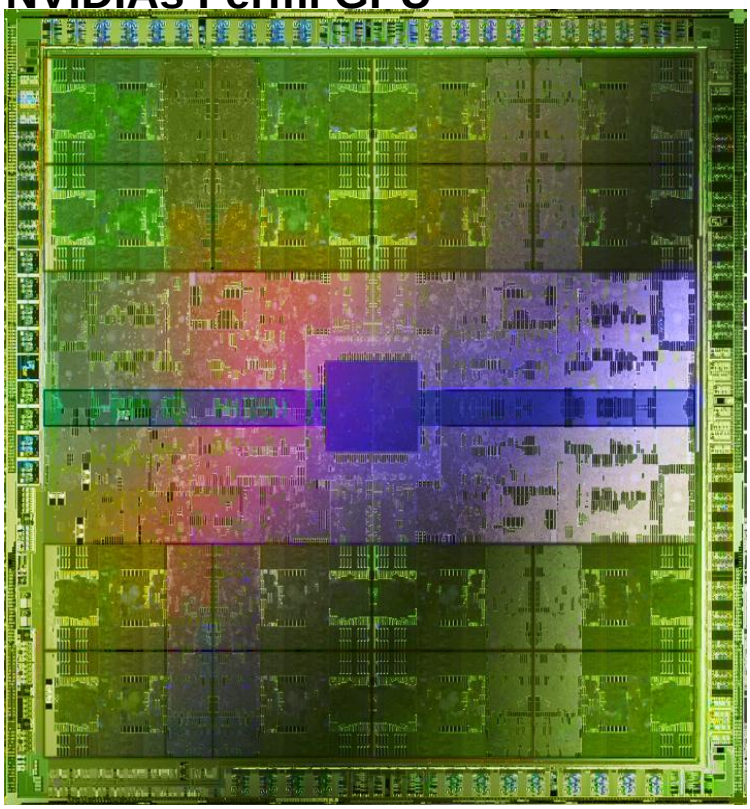
- 32 cores form a SM Streaming Multiprocessor
- 16 SIMD units
- ECC support Error Detection and Correction
- Unified Level 2 Cache
- Six 64-bit memory partitions (384 bit memory interface)

NVIDIAs Fermi GPU

- 512 CUDA Cores (Compute Unified Device Architecture)
- Parallel computing architecture developed by NVIDIA
- Ececutes one floating point and one integer instruction per clock
- 16 Load / Store Units per SM
- SFU special function unit handles sine, cosine, square root, interpolation etc.
- 64KB Level 1 Cache

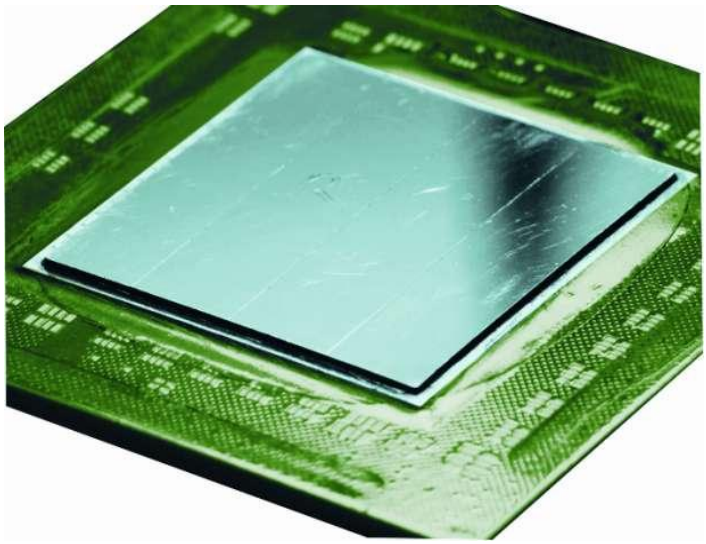


NVIDIAs Fermi GPU



- General Purpose Graphics Processing Unit (GPGPU)
- spring **2011**
  largest chip ever built
- 3 Billion Transistors
- 512 Cores

Xilinx Virtex-7 LX2000T



Frühjahr **2012**

2.5D Stacked Silicon Interconnect (SSI) technology

4 Assembly optimized FPGA Slices

6.8 Billion Transistors (28nm)

2 Million Logic Cells (CLB)
(10 Million Gates)
- Real 6-input look-up tables (LUTs)
- Memory capability within the LUT
- Register and shift register functionality

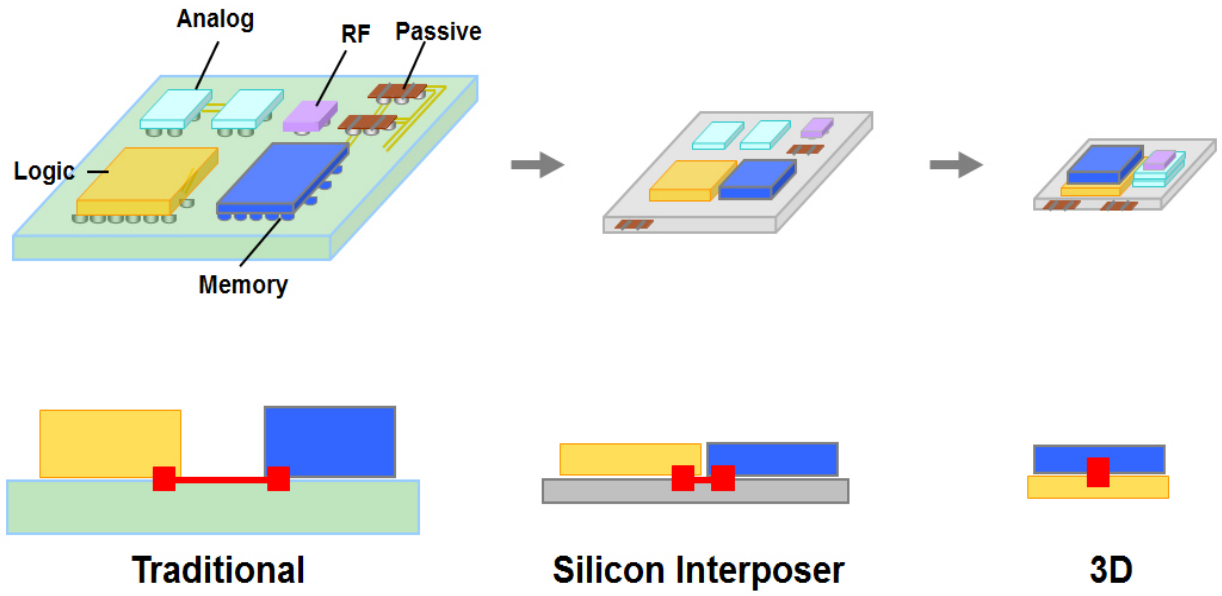
Dual-port 36 Kb block RAM
with port widths of up to 72
built-in optional error correction circuitry
built-in FIFO controller

DSP Slice
25 × 18 two's complement multiplier/accumulator
high-resolution (48 bit) signal processor

I/O Gb/s serial transceivers
6.6 Gb/s, 12.5 Gb/s, 13.1 Gb/s or 28.05 Gb/s

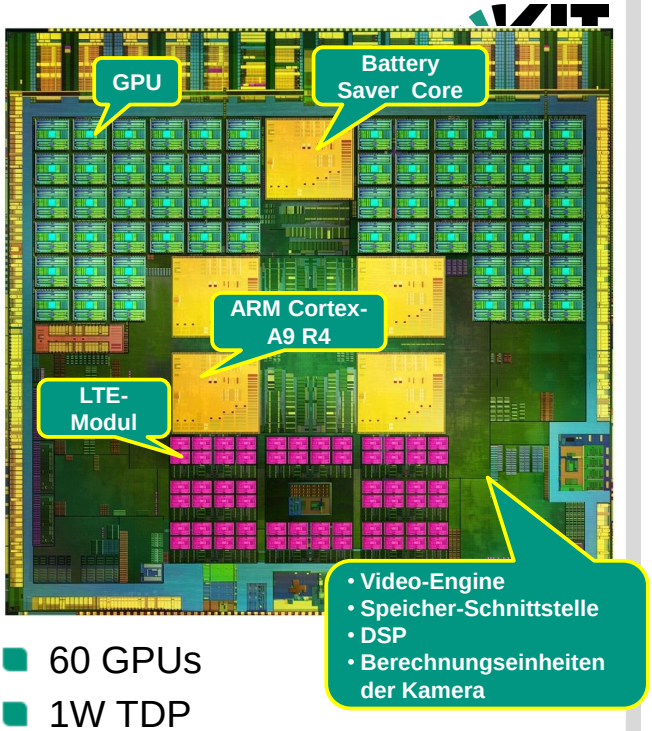
SelectI/O technology
with support for 1,866 Mb/s DDR3

System Migration Trend



Nvidia Tegra 4i

- 1 ARM Cortex-A9
  - „Battery Saver Core“
  - Optimiert auf Low-Power
    - Eigener Cache, 512 kB
- 4 ARM Cortex-A9 R4
  - 0, 1, 2 oder 4 aktiv
  - Cache reallokierbar
  - Verbesserte Eigenschaften:
    - Optimiert auf Leistung
    - 32 kB L1-Daten-Cache
    - L1-Prefetcher und Prefetch-Puffer
    - Größere TLB (512 Einträge)
    - Überarbeitete Sprungvorhersage
- LTE-Modem integriert
  - Software-Defined-Radio
  - 8 programmierbare Einheiten

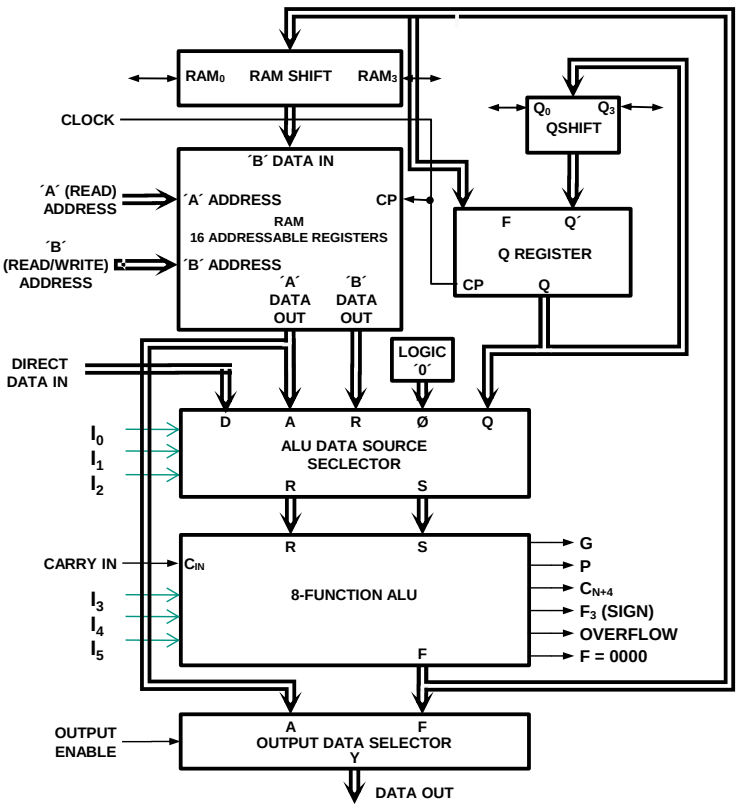


- 60 GPUs
- 1W TDP
- In 28nm gefertigt

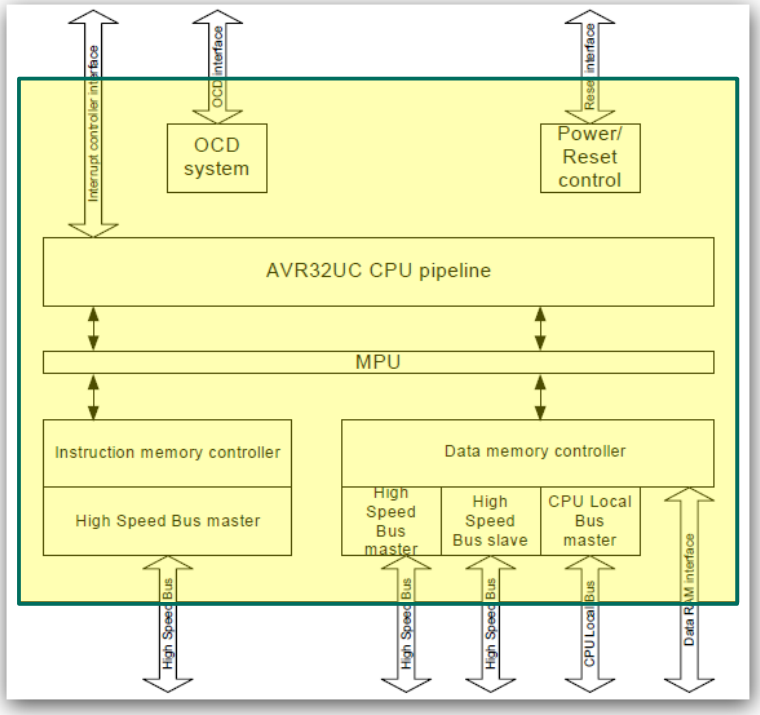
Frühjahr **2013**

Beispiel-Architekturen

4-Bit Arithmetik-Logik-Einheit
(AMD 2901 Slice)



ATMEL CPU (Workshop)



ATMEL 32UC3B256 Key Features

- Low Power 32 Bit RISC Architektur
- 7 Direct Memory Access (DMA) Kanäle
- Embedded Memory
  - 256kB Flash
  - 32kB SRAM
- Interrupt Controller
- Power Management

Device		Embedded SRAM	Embedded Flash	USB Data	HSB-PB Bridge A	HSB-PB Bridge B
Start Address		0x0000_0000	0x8000_0000	0xD000_0000	0xFFFF_0000	0xFFFE_0000
Size	AT32UC3B0512 AT32UC3B1512	96 Kbytes	512 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes
	AT32UC3B0256 AT32UC3B1256	32 Kbytes	256 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes
	AT32UC3B0128 AT32UC3B1128	32 Kbytes	128 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes
	AT32UC3B064 AT32UC3B164	16 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes

- USB (Universal Serial Bus) Support
- 3-kanaliger 16 Bit Timer/Counter (TC)
- 7-kanaliger 20 Bit Pulse Width Modulation (PWM) Controller
- 3 Universal Synchronous/Asynchronous Receiver/Transmitters (USART)
- Master/Slave Serial Peripheral Interface (SPI)
- 8-kanaliger 10 Bit Analog-Digital-Wandler
- 16 Bit Stereo Audio Bitstream Digital-Analog-Wandler
- JTAG Debug Schnittstelle
- …

ATMEL AVR32 CPU Key Features

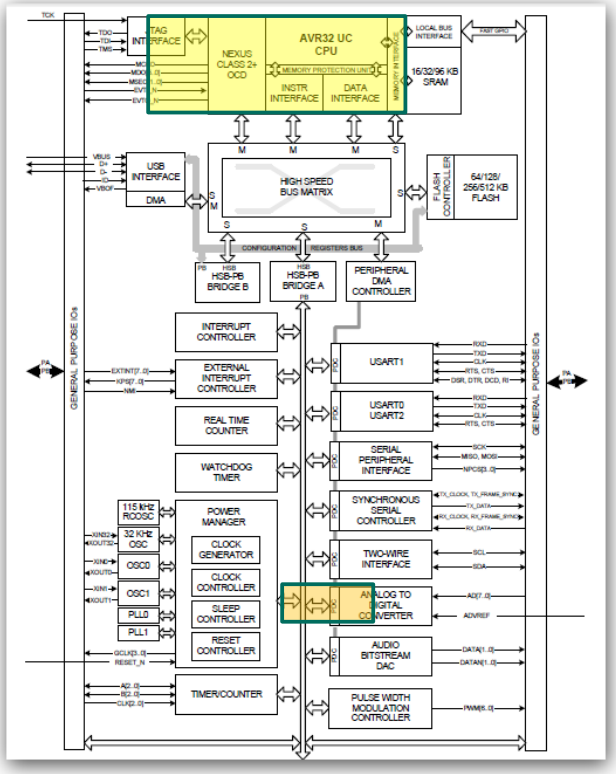
- 32 Bit Load/Store RISC Architektur
- 15 General Purpose 32 Bit Register
- 32 Bit Stack Pointer, Program Counter und Link Register
- 3-stufige Pipeline
- Byte, Half-Word, Word und Double Word Speicherzugriff
- Fast Interrupts mit 4 Interrupt Prioritäts-Ebenen
- DSP Erweiterungen
  - Sättigungsarithmetik
  - Multiplikationen
- Registerbezeichnungen:
  - PC: Program Counter
  - LR: Link Register
  - SP_APP: Stack Pointer (for Applications)
  - SR: Status Register
  - R0..12: Register
    - Enthalten die Operanden

Bit 31	Bit 0
PC	
LR	
SP_APP	
R12	
R11	
R10	
R9	
R8	
R7	
R6	
R5	
R4	
R3	
R2	
R1	
R0	
SR	

ATMEL Microcontroller (Workshop)

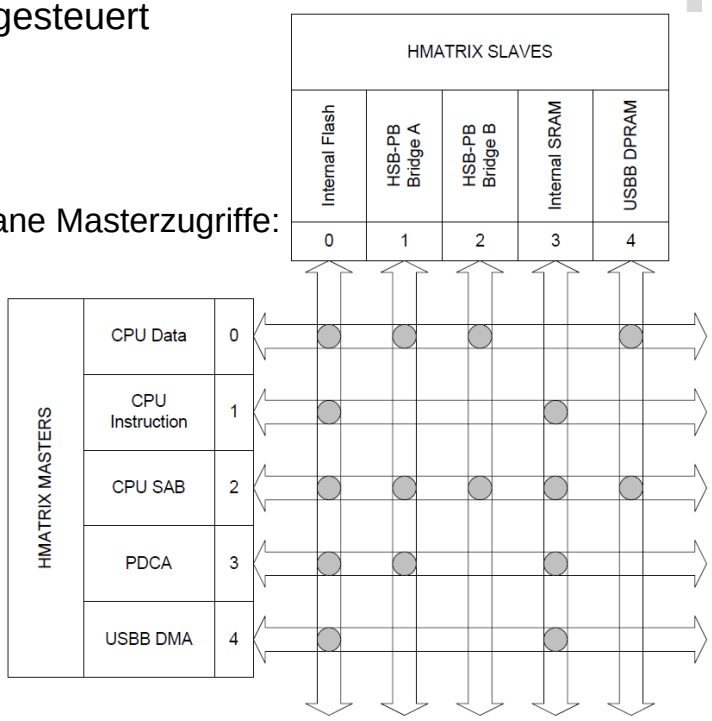
System-on-Chip

Peripherie für
Messen,
Steuern,
Regeln



High Speed Bus Matrix, unterstützt DMA

- Wird als normale Peripherie angesteuert
  - Adresse: 0xFFFE‘1000
- Programmierbar über Register:
  - Bis zu 16 Master und Slaves
  - 2 Arbitrierungsarten für simultane Masterzugriffe:
    - Round Robin
    - Fixed Priority
  - Prioritäten der Master können für jeden Slave einzeln definiert werden
- Latenzen:
  - 1 Takt für ersten Zugriff eines Bursts
  - 0 Takte für Default-Master



Instruction Set Architecture ISA

- Die Sicht des Programmierers auf den Computer hängt ab von der Antwort auf fünf Fragen:
  - Wie werden Daten repräsentiert?
  - Wo können Daten gespeichert werden?
  - Wie kann auf Daten zugegriffen werden?
  - Welche Operationen können mit den Daten ausgeführt werden?
  - Wie werden Befehle codiert?
  - Die Antwort auf diese Fragen definiert die externe Architektur oder Befehlssatzarchitektur
    Englisch: Instruction Set Architecture (ISA)

Datenrepräsentation Datenformate

- ISA unterstützt verschiedene Datenformate
  - Integer (z.B. Byte, 16-bit Word, 32-bit Longword, and 64-bit Quadword
  - Es gibt zwei Arten Byte-Adressen in einem Wort zu ordnen:
    - big-endian: most significant byte first und
    - little-endian: least significant byte first
  - Gepackte und ungepackte BCD Zahlen
  - ASCII Character
  - Floating-point data formats (ANSI/IEEE 754-1985):
    standard, basic oder extended, je zwei Längen: single or double
  - Multimedia: 32-, 64- oder 128-bit Worte, 8- or 16-bit pixel Farbtiefe.

Datenrepräsentation Datenformate – AVR32

- ISA unterstützt verschiedene Datenformate
  - Integer
    - Byteweise Adressierung im Speicher
    - Zugriff kann in den folgenden Bitbreiten durchgeführt werden:
      - Byte (8 bit)
      - Halfword (16 bit)
      - Word (32 bit)
      - Double word (64 bit)
    - Zweierkomplement bei signed und Floating Point.
  - Intern benutzte Byte-Adressierung: big-endian
    - Um den Datentransfer mit little-endian-Adressierung zu ermöglichen, existieren spezielle Load-/Store-Operationen zur Übersetzung der Endianness
  - Floating-point data formats (ANSI/IEEE 754-1985):
    Die FP-Hardware ist konform mit den Anforderungen des C-Standards.
- Die AVR32 ISA besteht aus 16-bit (compact) und 32-bit (extended) breiten Instruktionen.

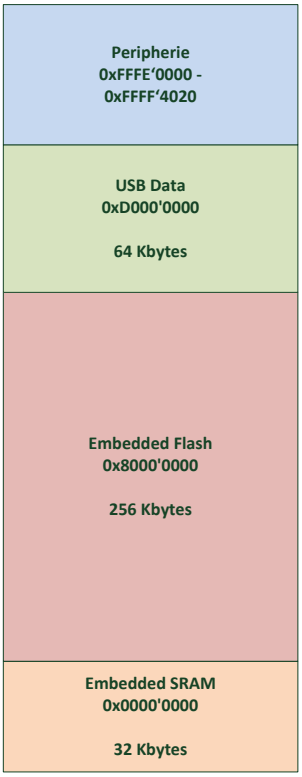
Datenspeicherung - Adressraum

- Adressraum Assemblerebene:
  register space, stack space, I/O space, control space
- Außer Register werden alle anderen Adressräume auf einen einzigen zusammenhängenden Speicheradressraum abgebildet
- Eine Reduced Instruction Set (RISC) ISA enthält zusätzlich einen relativ großen Registersatz (general-purpose CPU registers)
- Heutige RISC Prozessoren zusätzlicher Satz (z.B. 32) 64-bit floating-point Register.

Weitere Details ab Folie 12-121

Datenspeicherung – Adressraum – AVR

- Zwei separate Speicherbereiche:
  - Register Space
  - Speicher und Peripherie-Geräte werden über einen gemeinsamen Adressraum verwaltet
- Beispiele für Peripheriegeräte:
  - DMA-Controller
  - AD-Wandler
- Zugriffsschutz wird per Memory-Protection-Unit (MPU) geregelt
  - Z.B. Geschützter Bereich für das Betriebssystem
- Register:
  - 15 General-Purpose 32-bit-Register
  - 16 Register im Register-File
    - Program Counter, Link Register, Stack Pointer
  - Viele weitere Register für Steuerung Peripherie



Datenzugriff - Adressierungsmodi (1)

Register	load Reg1,Reg2 Reg1 ← (Reg2)
Immediate	load Reg1,#const Reg1 ← const
Direct	load Reg1,(const) Reg1 ← Mem[const]
Register indirect	load Reg1,(Reg2) Reg1 ← Mem[(Reg2)]

Datenzugriff - Adressierungsmodi (2)

Post-Increment	load Reg1,(Reg2)+ Reg1 ← Mem[(Reg2)], Reg2 ¬ (Reg2) + step
Pre-Decrement	load Reg1,-(Reg2) Reg2 ← (Reg2) - step, Reg1 ¬ Mem[(Reg2)]
Displacement	load Reg1,displ(Reg2) Reg1 ← Mem[displ + (Reg2)]
Indexed and scaled indexed	load Reg1,(Reg2*scale) Reg1 ← Mem[(Reg2)*scale]

Datenzugriff - Adressierungsmodi (3)

Indirect scaled indexed	load Reg1,(Reg2,Reg3*scale) Reg1 ← Mem[(Reg2) + (Reg3)*scale]
Indirect scaled indexed with displacement	load Reg1,displ(Reg2,Reg3*scale) Reg1 ← Mem[displ + (Reg2) +(Reg3)*scale]
PC-relative	branch displ PC ← PC + step + displ (if branch taken)

Datenzugriff - Adressierungsmodi – AVR32

- LD.W – Load Word
  - Rp: Register enthält Pointer auf Quell-Speicheradresse
  - Rd: Ziel-Register
  - Adressierungsarten werden anhand der unterschiedlichen Syntax gesteuert

- Verschiedene Adressierungsarten:
  - Post-inkrement
    - Syntax: ld.w Rd, Rp++
    - Semantik: Rd := *(Rp);
      Rp := Rp + 4;
  - Pre-dekrement
    - Syntax: ld.w Rd, --Rp
    - Semantik: Rp := Rp – 4;
      Rd := *(Rp);

Register mode: the operand is stored in one of the registers.

Immediate (or literal) mode: the operand is a part of the instruction.

Direct (or absolute) mode: the address of the operand in memory is stored in the instruction.

Register indirect (or register deferred) mode: the address of the operand in memory is stored in one of the registers.

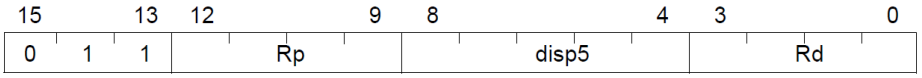
Autoincrement (or register indirect with postincrement) mode: like the register indirect, except that the content of the register is incremented after the use of the address. This mode offers automatic address increment useful in loops and in accessing byte, halfword, or word arrays of operands.

Autodecrement (register indirect with predecrement) mode: the content of the register is decremented and is then used as a register indirect address. This mode can be used to scan an array in the direction of decreasing indices.

Datenzugriff - Adressierungsmodi – AVR32

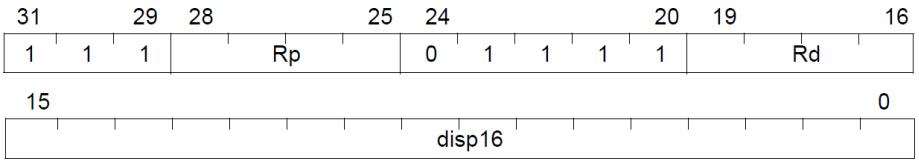
- Verschiedene Adressierungsarten:
  - Displacement (16-bit-Format)
    - Syntax: ld.w Rd, Rp[disp]
    - Semantik: Rd := *(Rp + (ZE(disp5) << 2);

Format III:



  - Displacement (32-bit-Format)
    - Syntax: ld.w Rd, Rp[disp]
    - Semantik: Rd := *(Rp + (SE(disp16)));

Format IV:



(nähere Informationen in doc32000.pdf)

Displacement (also register indirect with displacement or based) mode: the effective address of the operand is the sum of the contents of a register and a value, called displacement, specified in the instruction.

Indexed and scaled indexed mode: works essentially as the register indirect. The register containing the address is called index register.

The main difference between the register indirect and the indexed is that the contents of the index register can be scaled by a scale factor (e.g. 1, 2, 4, 8 or 16).

The availability of the scale factor, along with the index register, permits scanning of data structures of any size, at any desired step.

Indirect scaled indexed mode: the effective address is the sum of the contents of the register and the scaled contents of the index register.

Indirect scaled indexed with displacement mode: essentially as the indirect scaled indexed, except that a displacement is added to form the effective address.

Operationen mit Daten

- Datentransfer
- Arithmetisch
- Floating Point
- Logisch
- Bedingte und unbedingte Verzweigung
- Shift, Rotate
- Bit Manipulation
- Multimedia
- (Systemsteuerung, Coprozessorbefehle)

Je nach Befehlsart vom Typ
register-register, memory-register,
register-memory, oder memory-memory.

PC-relative mode: a displacement is added to the PC. The PC-relative mode is often used with branches and jumps.

Operationen mit Daten – AVR32

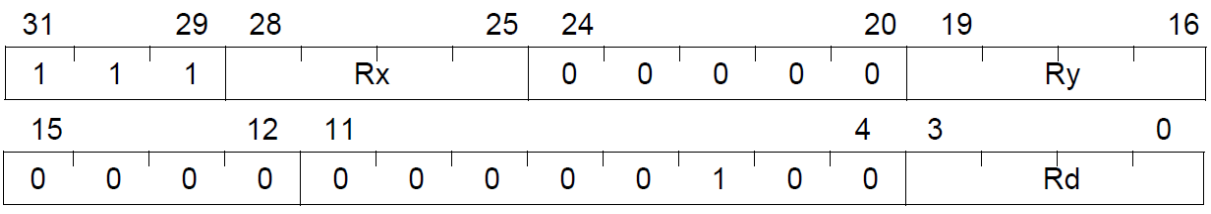
- Arithmetisch
  - adc: Add with carry
  - cp.w: wortweiser Vergleich
- Multiplication (separater Multiplizierer)
  - mac: multiply accumulate
- Logisch
  - eor: exklusives ODER
  - com: 1er-Komplement
- Bedingte und unbedingte Verzweigung
  - rjmp: relativer Sprung
- Shift, Rotate
  - asr: arithmetisches Schieben nach rechts mit Vorzeichen
- Bit Operationen
  - clz: zähle die führenden Nullen

Befehlsformate

- 3-Adress Befehlsformat: opcode | Dest | Src1 | Scr2
  typische Verwendung für Register-Register Befehl
  (Load/Store Machines)
- 2-Adress Befehlsformat: opcode | Dest/Src1 | Src2
  (Register-Memory Machines)
- 1-addressBefehlsformat: opcode | Src
- (Accumulator Machines)
- 0-address Befehlsformat: nur Befehlscode
  (Stack Machines)

Befehlsformate – AVR32

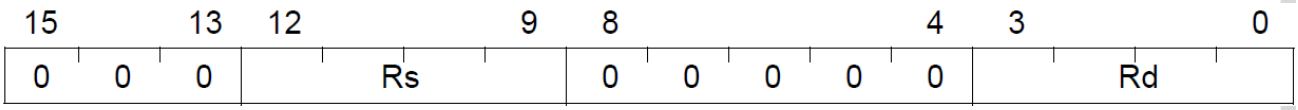
- 3-Adress Befehlsformat: opcode | Dest | Src1 | Scr2
- Beispiel: ADC – Add with Carry
  - Syntax: adc Rd, Rx, Ry
  - Semantik: Rd := Rx + Ry + C;



Befehlsformate – AVR32

- 2-Adress Befehlsformat: opcode | Dest/Src1 | Src2
  (Register-Memory Machines)

- Beispiel: ADD – Add without carry
  - Syntax add Rd, Rs
  - Semantik Rd := Rd + Rs;

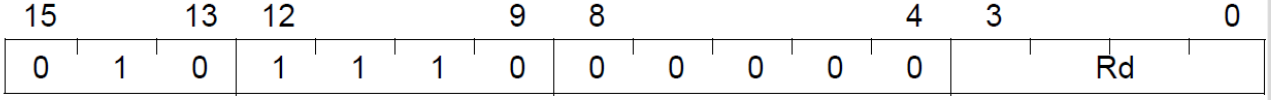


Befehlsformate – AVR32

- 1-Adress Befehlsformat: opcode | Src

- Beispiel: ACR – Add carry to register
  - Syntax acr Rd
  - Semantik Rd := Rd + C;

Opcode:

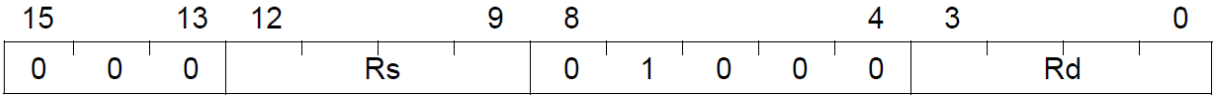


Befehlsformate – AVR32

- 2-Adress Befehlsformat: opcode | Dest/Src1 | Src2
  (Register-Memory Machines)

- Beispiel: ANDN – Logical AND NOT
  - Syntax andn Rd, Rs
  - Semantik Rd := Rd ^ ¬Rs;

Opcode(s):

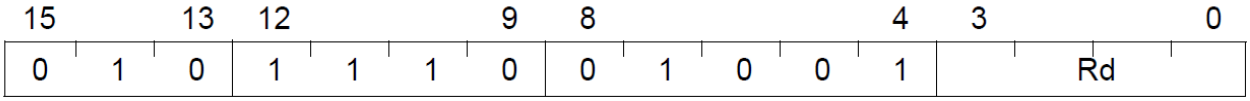


Befehlsformate – AVR32

- 1-Address Befehlsformat: opcode | Src

- Beispiel: BREV – Bit Reverse
  - Syntax brev Rd
  - Semantik Rd[31:0] := Rd[0:31];

Opcode:



Befehlsformate – AVR32

- 0-address Befehlsformat: nur Befehlscode

- Beispiel: NOP – No Operation
  - Syntax nop
  - Wird z.B. verwendet, um Wartezyklen beim Hardware-Zugriff zu realisieren.

Opcode:

15						9	8				4	3			0
1	1	0	1	0	1	1	1	0	0	0	0	0	0	1	1

ATMEL Microcontroller (Workshop)

Adresse	Gerät	Name
0xFFFE0000	USB	USB 2.0 Interface - USB
0xFFFE1000	HMATRIX	HSB Matrix - HMATRIX
0xFFFE1400	HFLASHC	Flash Controller - HFLASHC
0xFFFF0000	PDCA	Peripheral DMA Controller - PDCA
0xFFFF0800	INTC	Interrupt controller - INTC
0xFFFF0C00	PM	Power Manager - PM
0xFFFF0D00	RTC	Real Time Counter - RTC
0xFFFF0D30	WDT	Watchdog Timer - WDT
0xFFFF0D80	EIM	External Interrupt Controller - EIM
0xFFFF1000	GPIO	General Purpose Input/Output Controller - GPIO
0xFFFF1400	USART0	Universal Synchronous/Asynchronous Receiver/Transmitter - USART0
0xFFFF1800	USART1	Universal Synchronous/Asynchronous Receiver/Transmitter - USART1
0xFFFF1C00	USART2	Universal Synchronous/Asynchronous Receiver/Transmitter - USART2
0xFFFF2400	SPI0	Serial Peripheral Interface - SPI0
0xFFFF2C00	TWI	Two-wire Interface - TWI
0xFFFF3000	PWM	Pulse Width Modulation Controller - PWM
0xFFFF3400	SSC	Synchronous Serial Controller - SSC
0xFFFF3800	TC	Timer/Counter - TC
0xFFFF3C00	**ADC**	**Analog to Digital Converter - ADC**
0xFFFF4000	ABDAC	Audio Bitstream DAC - ABDAC

- Analog-Digital-Converter
  - Base address: 0xFFFF3C00
  - Size (in bytes): **0x400** = 0xFFFF4000 – 0xFFFF3C00 = 1024

ATMEL Microcontroller (Workshop)

Offset	Register	Name	Access	Reset State
0x00	Control-Register	CR	Write-only	0x00000000
0x04	Mode-Register	MR	Read/Write	0x00000000
0x10	Channel-Enable-Register	CHER	Write-only	0x00000000
0x14	Channel-Disable-Register	CHDR	Write-only	0x00000000
0x18	Channel-Status-Register	CHSR	Read-only	0x00000000
0x1C	**Status-Register**	**SR**	**Read-only**	**0x000C0000**
0x20	**Last-Converted-Data-Register**	**LCDR**	**Read-only**	**0x00000000**
...	...	...	...	...
0xFC	Version-Register	VERSION	Read-only	Device-specific

- Status-Register (SR)
  - Enthält Status-Informationen über die Puffer des Wandlers
  - Bit 16 zeigt an, ob neue Daten vom Wandler erzeugt wurden
- Last-Converted-Data-Register (LCDR)
  - Speichert den Wert der letzten AD-Wandlung
  - wird nach jeder Konvertierung überschrieben

- Direkter Zugriff auf Register mit Zeigern:
  - ADC.Register-Adresse = ADC.Basis-Adresse + ADC.Offset

ATMEL Microcontroller (Workshop)

25.7.6 Status Register

Name: SR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x000C0000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- DRDY: Data Ready
  - Bit 16 im Register → 17. Stelle von rechts
  - Dieses Bit wird gesetzt, wenn ein neuer Wert im Register LCDR verfügbar ist. Es wird gelöscht, wenn LCDR gelesen wird.

ATMEL Microcontroller (Workshop)

Offset	Register	Name	Access	Reset State
0x00	Control-Register	CR	Write-only	0x00000000
0x04	Mode-Register	MR	Read/Write	0x00000000
0x10	Channel-Enable-Register	CHER	Write-only	0x00000000
0x14	Channel-Disable-Register	CHDR	Write-only	0x00000000
0x18	Channel-Status-Register	CHSR	Read-only	0x00000000
0x1C	**Status-Register**	**SR**	**Read-only**	**0x000C0000**
0x20	**Last-Converted-Data-Register**	**LCDR**	**Read-only**	**0x00000000**
...	...	...	...	...
0xFC	Version-Register	VERSION	Read-only	Device-specific

- ADC
  - Basis-Adresse: 0xFFFF3C00
  - Neuer Wert in LCDR: Bit 16
- Beispiel: Status-Register auswerten (neuer Wert verfügbar?)

```
uint32_t* adc_basis = 0xFFFF3C00; // Zeiger auf den Speicher wg. *
uint32_t sr_offset = 0x0000001C;
uint32_t sr_value = *( adc_basis + sr_offset );// * = Adresse auflösen
// Bit 16 (17. Bit von rechts) ausmaskieren
uint32_t neuer_wert = sr_value & 0x00010000;
// wenn neuer_wert nicht Null ist, ist steht ein neuer Wert in LCDR!
```


ATMEL Microcontroller (Workshop)

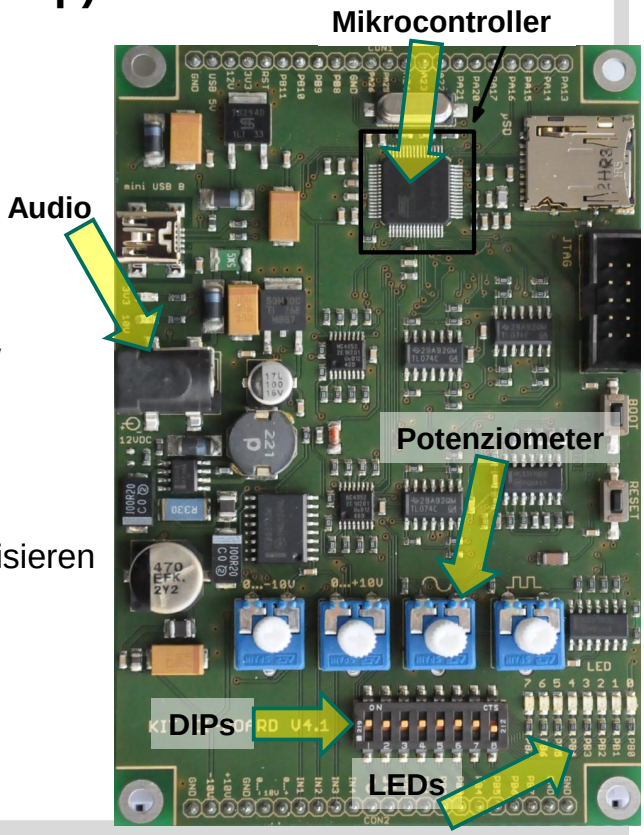
- LCDR auslesen, wenn neuer Wert verfügbar:

```
uint32_t* adc_basis = 0xFFFF3C00;
uint32_t sr_offset = 0x0000001C;
uint32_t lcdr_offset = 0x00000020;
uint32_t sr_value, lcdr_value, neuer_wert;
...
while(1) {   // typisch fuer "embedded": das Programm soll immer laufen
   ...
   sr_value = *( adc_basis + sr_offset );// * = Adresse auflösen
   neuer_wert = sr_value & 0x00010000;

   // wenn neuer_wert nicht Null ist, steht ein neuer Wert in LCDR!
   if ( neuer_wert ) {
      lcdr_value = *( adc_basis + lcdr_offset );// * = Adresse auflösen
      // tue etwas mit lcdr_value
      ...
   }
   ...
};
```

Weitere Möglichkeiten (Workshop)

- LEDs ansteuern
- DIP-Switches abfragen
  - Steuerung der Software
  - Bsp.: Zustandssteuerung
- Potenziometer abfragen
  - AD-Wandlung der Spannungen
  - Steuerung der Hardware per SW
  - Bsp.: Helligkeit der LEDs regeln
    - Durch Puls-Weiten-Modulation
- Audio-Buchse als Eingang
  - Bsp.: Lautstärke mit LEDs visualisieren



Weitere Beispiele ARM, IA64

- Nicht prüfungsrelevant

Beispiel ARM Prozessoren
Was ist ARM?

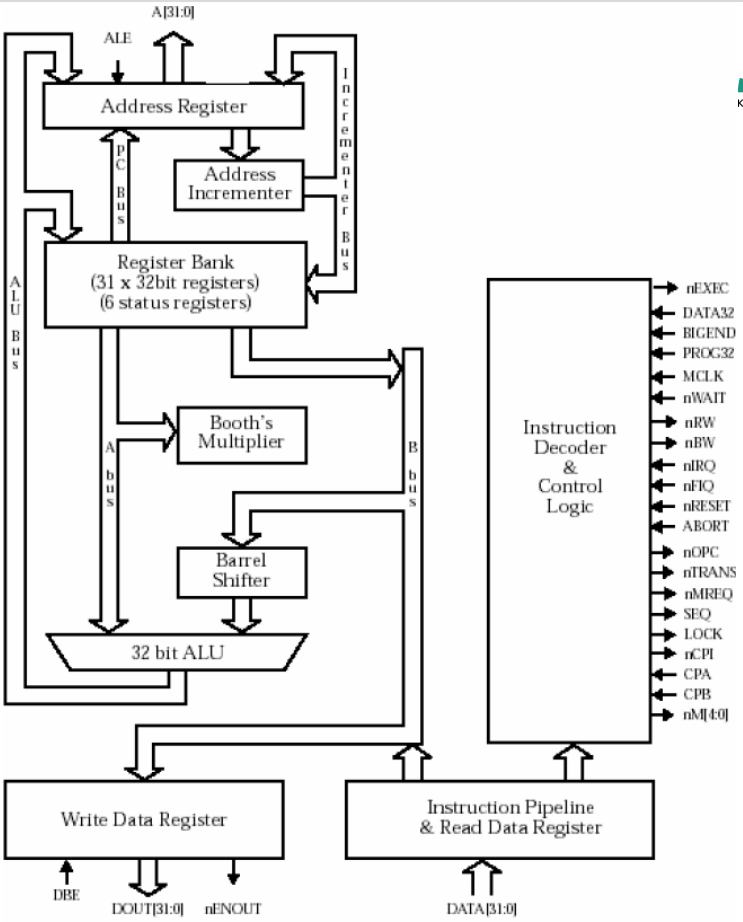
- Eine Firma
  - Acorn RISC Machine
  - Advanced RISC Machine
  - Advanced RISC Machines, Ltd.
- Ein Prozessordesign (Inhalt dieser VL)
- Bezeichnung von Prozessorkernen

ARM Prozessor-Varianten

- T-Variante:
  Unterstützt Thumb-Modus
- M-Variante:
  Unterstützt Multiplikation zweier 32-Bit-Zahlen zu einer 64-Bit-Zahl (lange Multiplikation)
- E-Variante:
  Enhanced DSP Instruktionen für Beschleunigung typischer DSP-Operationen
- J-Variante:
  Jazelle bietet Soft- und Hardwarebeschleunigung für Java-Bytecode

Mehr als 10 Versionen (Erweiterungen Architektur / Befehlssatz) seit 1993

ARM 7 Blockdiagramm



ARM Architektur Merkmale

- RISC-Design
- 32-Bit Adress- und Datenbus
- Load/Store-Architektur
- 3-Adress-Instruktionen mit fester Länge
- 7 Prozessor-Modi
- 37 Register
- Datentypen:
  - Wort (32 Bit), auf 4-Byte-Grenzen anliegend
  - Halbwort (16 Bit), auf 2-Byte-Grenzen anliegend
  - Byte (8 Bit)
- Instruktionen
  - 54 ARM-Instruktionen: genau ein Wort breit
  - 38 Thumb-Instruktionen: genau ein Halbwort breit
  - Datenverarbeitende Operationen arbeiten alle auf Wörtern

ARM Architektur

Prozessor-modus	Kurzform	Beschreibung	Exception
User	usr	Normale Programmausführung	-
System	sys	Privilegierte Programmausführung	-
Supervisor	svc	Privilegierter Exception-Modus	Software Interrupt
Abort	abt		Data Abort, Prefetch Abort
Undefined	und	Fängt undefinierte Instruktionen und unbehandelte Koprozessor-Instruktionen ab	Undefined Instruction
IRQ	irq	Wird für die Interruptbehandlung benutzt	IRQ
FIQ	fiq	Wird für die Interruptbehandlung FIQ benutzt (Fast Interrupt)	FIQ

Registersatz

- 37 Register
- R0 - R14 (meist) beliebig benutzbar
- Besondere Funktionen und Register:
  - R15: PC
  - R14: Link Register (Exception-Mode, BL-Instruktion)
  - R13: Stackpointer in Exception-Mode
  - CRSR: Current Rrogram Status Register
  - SRSR: Saved Program Status Register

Registersatz

User	System	Supervisor	Abort	Undefined	IRQ	FIQ
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_fiq
R9	R9	R9	R9	R9	R9	R9_fiq
R10	R10	R10	R10	R10	R10	R10_fiq
R11	R11	R11	R11	R11	R11	R11_fiq
R12	R12	R12	R12	R12	R12	R12_fiq
R13	R13	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14	R14	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

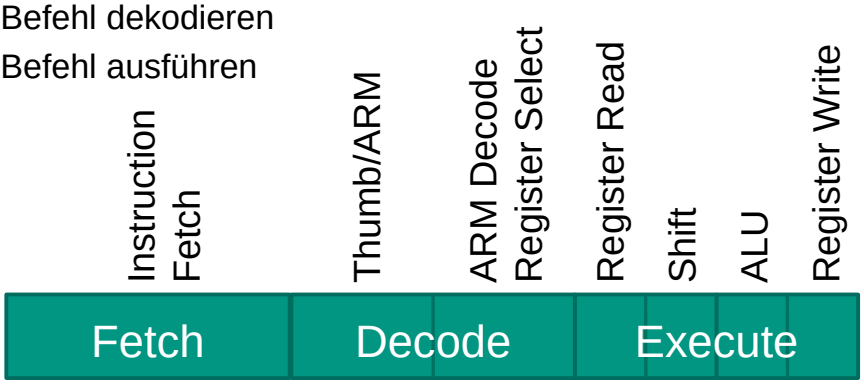
Statusregister

31	30	29	28	27	26 8	7	6	5	4	3	2	1	0
N	Z	C	V	Q	Undefiniert	I	F	T	M	M	M	M	M

- N (Negativ), Z (Null), C (Übertrag) und V (Überlauf) sind Condition-Flags
- Q: Nur in E-Variante als Überlauf für erweiterte DSP-Instruktionen
- I und F: Für Interrupt- bzw. Fast Interrupt-Mode
- T: Thumb-Mode
- M kodiert den Modus der CPU

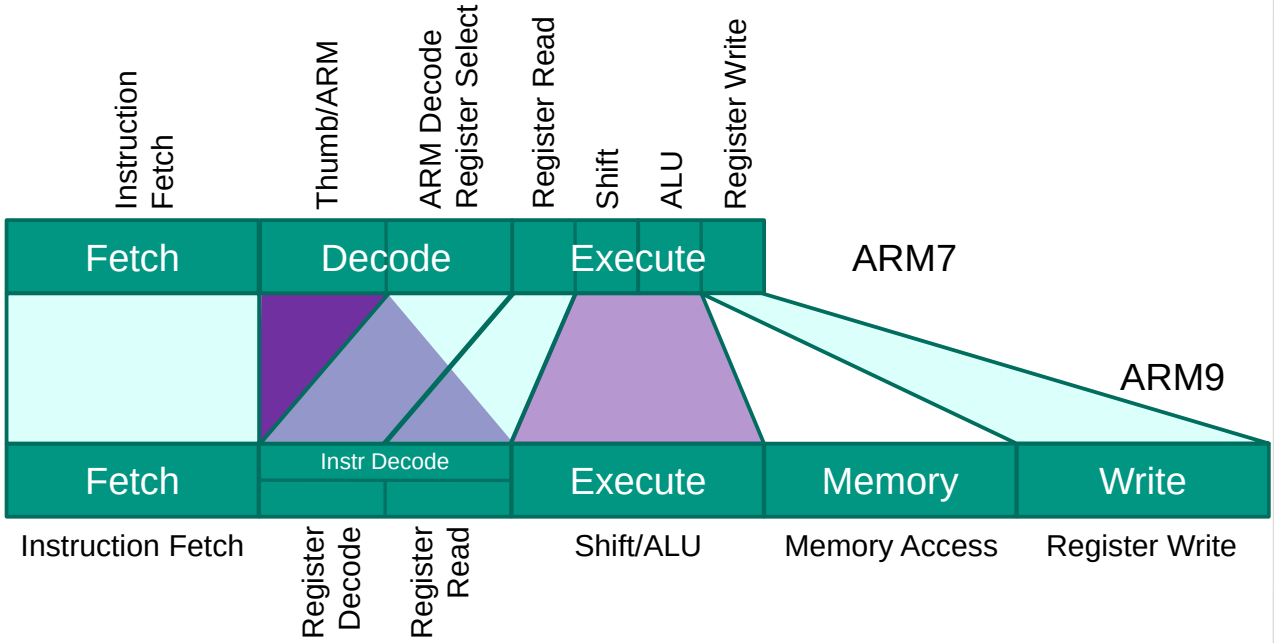
ARM 7 Pipeline Verarbeitung

- 3-stufige Pipeline
  - Befehl holen
  - Befehl dekodieren
  - Befehl ausführen



Stufe 1	Fetch	Decode	Execute	Fetch
Stufe 2		Fetch	Decode	Execute
Stufe 3			Fetch	Decode
	Takt	Takt	Takt	Takt

ARM 9 Prozessor Core
5 Pipeline-Stufen



ARM 9 Prozessor Core

- 5 Pipeline-Stufen
  - Fetch:
    - Holen des Befehls an Adresse (pc) aus dem Speicher
  - Decode
    - Decodierung der Instruktion
    - Lesen der Operanden aus der Registerbank;
  - ALU
    - Ausführen der Instruktion
    - Berechnen eines Ergebnisses (data processing)
    - Berechnen der Adresse für eine Lade-, Speicher- oder Verzweigungsoperation
    - Multizyklusoperationen: Multiply, Register-Schiebe-Op.
  - LS1
    - Laden oder Speichern der Daten
  - LS2
    - Null- oder Vorzeichenerweiterung bei Byte- oder Halbworten

Ausnahmen-Abarbeitung (exception handling)

- Wenn eine Exception auftritt, wird folgender Code ausgeführt:

```
R14_<exception_mode> = return link
SPSR_<exception mode> = CPSR
CPSR[4:0] = exception mode number
CPSR[5] = 0             ; Execute in ARM state
If <exception_mode> = RESET or FIQ
    CPSR[6] = 1         ; Disable FIQs
CPSR[7] = 1             ; Disable IRQs
PC = exception vector address
```


Ausnahmen-Abarbeitung

- Exception Vector

Exception	Modus	EV-Adresse	EV-Adresse (High Vector)
Reset	Supervisor	0x00000000	0xFFFF0000
Undefined Instruction	Undefined	0x00000004	0xFFFF0004
Software Interrupt (SWI)	Supervisor	0x00000008	0xFFFF0008
Prefetch Abort (Speicher signalisiert Abbruch beim hohlen einer Instruktion)	Abort	0x0000000C	0xFFFF000C
Data Abort (Speicher signalisiert Abbruch beim hohlen eines Datenwortes)	Abort	0x00000010	0xFFFF0010
IRQ (interrupt)	IRQ	0xFFFF0018	0xFFFF0018
FIQ (fast interrupt)	FIQ	0x0000001C	0xFFFF001C

Ausnahmen-Abarbeitung

Priorität		Exception
höchste	1	Reset
	2	Data Abort
	3	FIQ
	4	IRQ
	5	Prefetch Abort
niedrigste	6	Undefined Instruction
	6	SWI

Interrupt Flags

- FIQ – Fast Interrupt reQuest
- IRQ – Interrupt reQuest

- Interrupts dienen dazu, laufende Programme zu unterbrechen
  - Um z.B. auf Signale der Peripherie (z.B. Maus oder Tastatur) schnell reagieren zu können.
  - Das Programm wir danach an der entsprechenden Stelle wieder fortgesetzt

Instruktionsmodi

- ARM-Mode
  - Standard-Modus
  - Einzig möglicher Modus für Ausnahme-Behandlung
  - 54 Instruktionen
- Thumb-Modus
  - Höhere Code-Dichte durch 16-Bit-Instruktionen
  - 38 Basis-Instruktionen
  - Weniger mächtig als ARM
  - Wechsel jederzeit möglich: BX (Branch-and-Exchange) bei gesetztem untersten Bit der Adresse
- Jazelle-Modus
  - Beschleunigte Ausführung von Java-Bytecode

ARM Instruktionstypen

- Branch-Instruktionen
- Datenverarbeitende Instruktionen
- Multiplikations-Instruktionen
- Statusregister-Instruktionen
- Load/Store-Instruktionen
- Load/Store-Multiple-Instruktionen
- Semaphor-Instruktionen
- Instruktionen, die Exceptions generieren
- Koprozessor-Instruktionen

ARM Instruktionen

- Alle Instruktionen sind 32 Bit lang
- Sehr strukturierte Kodierung: semantisch ähnliche Instruktionen werden auch ähnlich kodiert
- Bedingte Ausführung aller Instruktionen
  - Obere Bits (28-31) sind stets für die Bedingung reserviert
  - Keine Bedingung: AL (always)
  - Details: Nächste Folien
- Adressierungsmodi
  - Zwei oder drei Operanden (Ziel, Quelle, optionale Quelle)
  - Ziel und optionale Quelle immer Register
  - Quelle: "Shifter-Operand" kann innerhalb eines Zyklus um in der Instruktion angegebenen Wert geshiftet werden
  - Details: Nächste Folien

Instruktionssatz

- Allgemeine Instruktionsform:

```
<opcode>{<cond>}{S} <Rd>, <Rn>, <shifter_operand>
```

- Branch mit Link:

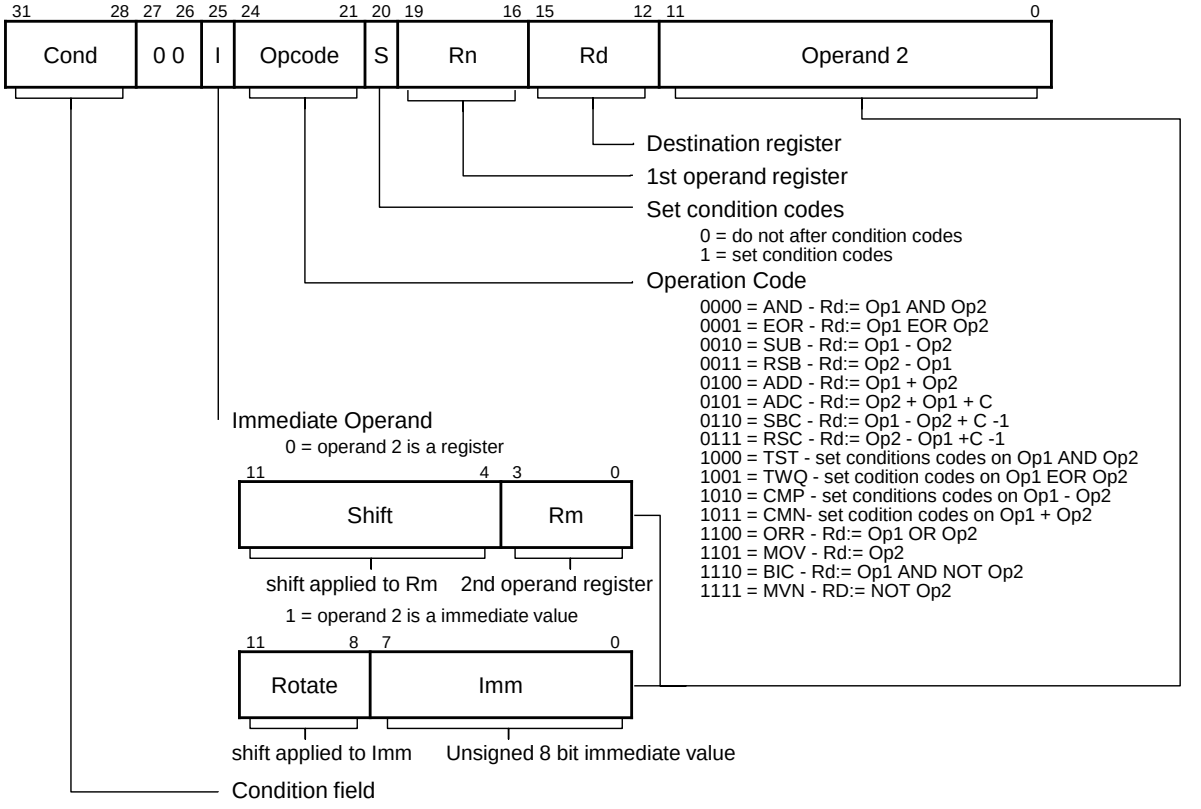
```
BL{<cond>} <target_address>
```

  - R14 : Adresse der Instruktion nach BL

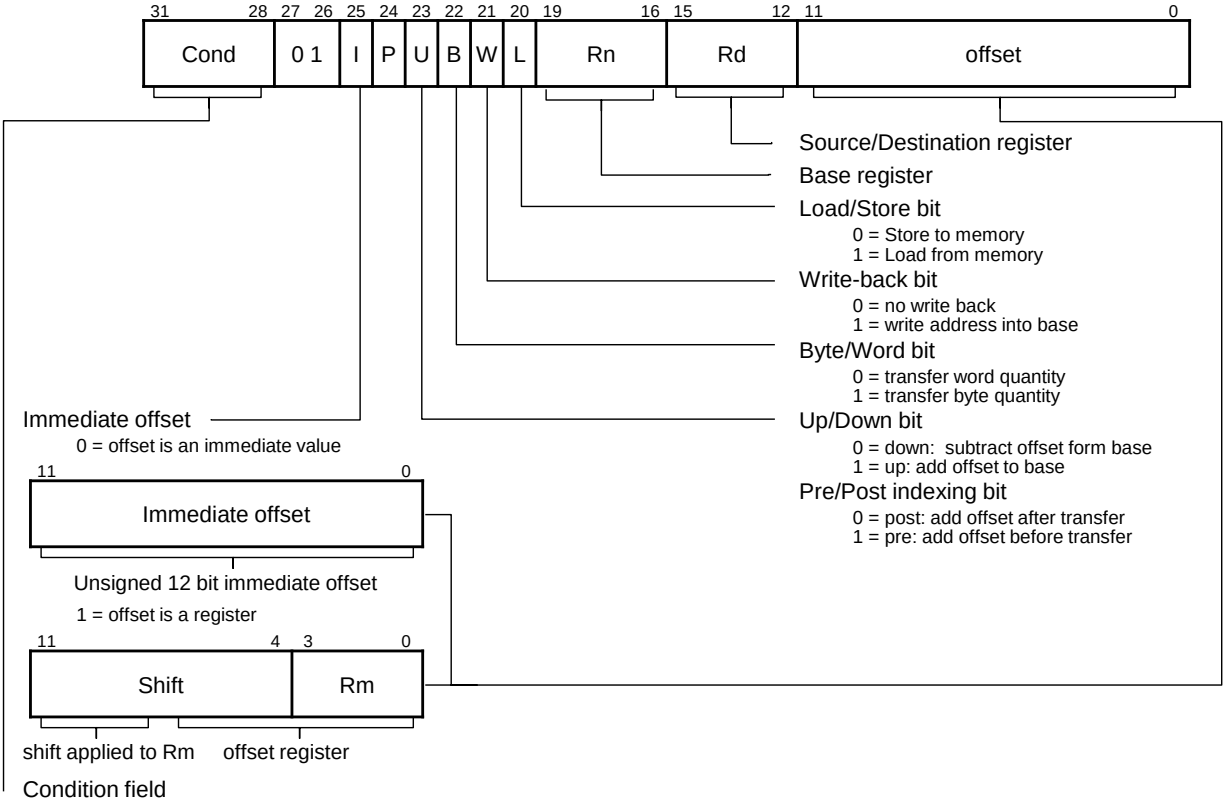
Instruktionsformat

31–28	27 26	25	24–21	20	19–16	15–12	11–8	7–5	4	3–0	
Cond	0 0	I	Opcode	S	Rn	Rd	Operand 2				Data Processing PSR Transfer
Cond	0 0 0 0 0 0			A	S	Rn	Rd	Rs	1 0 0 1	Rm	Multiply
Cond	0 0 0 1 0		B	0 0		Rn	Rd	0 0 0 0	1 0 0 1	Rm	Single Data Swap
Cond	0 1	I	P	U	B	W	L	Rn	Rd	offset	Single Data Transfer
Cond	0 1 1	XXXXXXXXXXXXXXXXXXX							1	XXXX	Undefined
Cond	1 0 0	P	U	S	W	L	Rn	Register List			Block Data Transfer
Cond	1 0 1	L	offset								Branch
Cond	1 1 0	P	U	N	W	L	Rn	CRd	CP#	offset	Coproc Data Transfer
Cond	1 1 1 0	OP Opc			CRn	CRd	CP#	CP	0	CRm	Coproc Data Operation
Cond	1 1 1 0	OP Opc		L	CRn	CRd	CP#	CP	1	CRm	Coproc Register Transfer
Cond	1 1 1 1	ignored by processor									Software Interuppt

Instruktionsformat



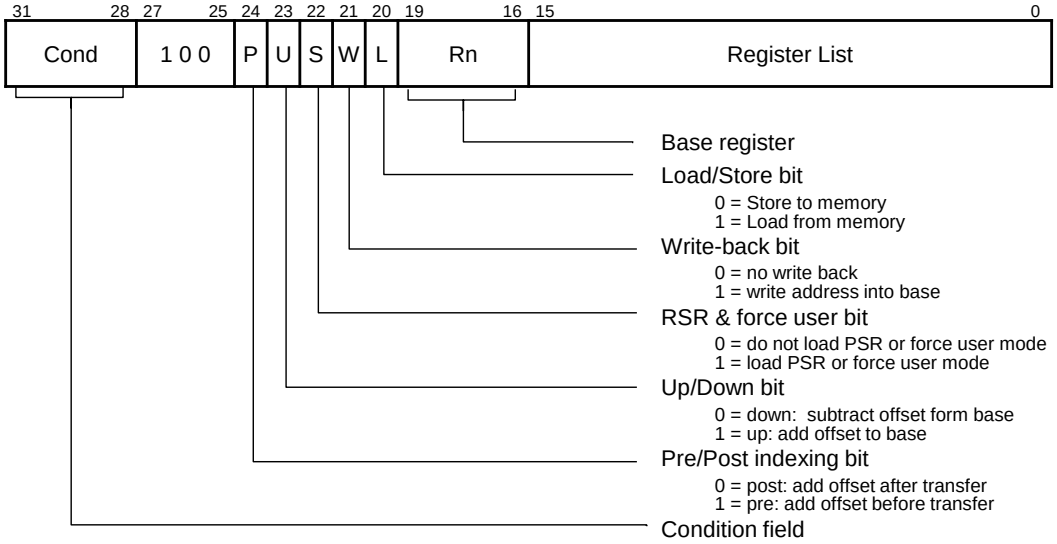
Single Data Transfer



Single Data Transfer

- Adressierungsarten: Indizierungsmethoden
  - Preindex with write back: mem[base+offset]
    - `ldr r0,[r1,#4]!`
    - Zugriff auf ein Element in einer Datenstruktur
  - Preindex mem[base+offset]
    - `ld1: 1:0,[1:1,#4]`
    - Traversieren von Feldern
  - Postindex mem[base]
    - `ldr r0,[r1],#4`
    - Traversieren von Feldern

Block Data Transfer



Load/Store Multiple

- Erlaubt das Laden von Registern mit dem Inhalt aufeinander folgender Speicheradressen bzw. das Speichern von Registern an aufeinander folgende Speicheradressen
- Mehrere Varianten
- Beispiel: STMDA R13!, {RO R12, LR}
  - STDMA Store Multiply Decrement After
  - Indexregister wird nach jedem Schritt dekrementiert
  - R13 ist Indexregister mit der Basisspeicheradresse
  - ! besagt, dass R13 nach jedem Schritt aktualisiert wird
  - RO-R12 und LR sind die Register, die in den Speicher kopiert werden sollen
  - Ablauf: RO nach Adresse, die in R13 steht, R13 um 4 dekrementieren, R1 an diese Adresse, usw.

Semaphore Instruktionen

- SWP (Swap) und SWPB (Swap Byte) erlauben atomare Lese/Schreibe-Operationen
- Beispiel 1:
  - SWP R1, R2, [R3]
  - R3 enthält Speicheradresse
  - SWP lädt R1 mit dieser Speicheradresse und schreibt R2 an diese Adresse
- Beispiel 2:
  - SWP R1, R1, [R2]
  - R2 enthält Speicheradresse
  - SWP tauscht Inhalt von R1 und der Speicheradresse

Thumb Instruction Set

- ARM7 Thumb Instruction Set
  - Thumb
    - Kodiert eine Teilmenge der 32-Bit ARM Befehle in einer 16-Bit Befehlsmenge
  - Verbesserung der Code-Dichte
    - In eingebetteten Systemen können die Kosten für RAM und ROM einen signifikanten Anteil an den Systemkosten haben
    - Höhere Code-Dichte (65% des ARM Codes)
    - Wichtiger Aspekt in speicherbeschränkten eingebetteten Systemen (z.B. Mobiltelefon, PDA)
  - Höhere Leistung, wenn mit einem 16-Bit Speicherport verbunden

Instruktionssatz Thumb I

- 16 Bit Instruktionslänge
- 38 Basis-Instruktionen
- Jederzeit Zugriff auf 8 Register (R0-R7)
- einzelne Instruktionen bieten auch Zugriff auf PC (R15), LR (R14), SP (R13) oder High Registers (R8-R15)
- Wechsel zur Laufzeit (mit BX, BLX, SRSR,LDR/LDM)
- Exceptions werden immer im ARM-Modus behandelt in jedem ARM-System existiert damit gemischter Code

Instruktionssatz Thumb II

- Branch-Instruktionen
- Datenverarbeitende Instruktionen
- KEINE Multiplikations-Instruktionen
- KEINE Statusregister-Instruktionen
- Load/Store-Instruktionen
- Load/Store-Multiple-Instruktionen
- KEINE Semaphor-Instruktionen
- Instruktionen, die Exceptions generieren
- KEINE Koprozessor-Instruktionen

- Instruktionsform:
  `<opcode> <Rd>, <Rn>, <3rd_operand>`
- 3. Operand: Register oder kurzer Immediate-Wert

Vergleich Code-Dichte

- ARM7 Thumb Insruction Set
  - Verbesserung der Code-Dichte

- ARM code

```
ARMDivide
; IN:   r0 (value), r1(divisor)
; OUT:  r2 (MODulus), r3(DIVide)

        MOVE    r3,#0
loop
        SUBS    r0,r0,r1
        ADDGE   r3,r3,#1
        BEG     loop
        ADD     r2,r0,r1
```

- Benötigter Speicherplatz: 5x4 Bytes

- THUMB code

```
THUMBDivide
; IN:   r0 (value), r1(divisor)
; OUT:  r2 (MODulus), r3(DIVide)

        MOVE    r3,#0
loop
        ADD     r3,#1
        SUB     r0,r1
        BEG     loop
        SUB     r3,#1
        ADD     r2,r0,r1
```

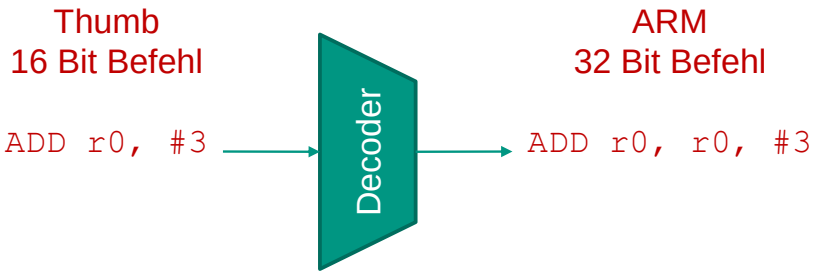
- Benötigter Speicherplatz: 5x4 Bytes

ARM 7 Thumb Instruction Set

Mnemonic	Description	ARM Equivalent
ADD ADC	Add Add with Carry	ADDS Rd, Rs, Rn ADCS Rd, Rd, Rs
SUB SBC	Substarct Substarct with Carry	SUBS Rd, Rs, Rn SBCS Rd, Rd, Rs
MUL AND	Multiply Logical AND	MUL Rd, Rs, Rd AND Rd, Rs, Rn
ORR EOR	Logical OR Exclusive-OR	ORRS Rd, Rd, Rs EORS Rd, Rd, Rs
LSL LSR	Logical Shift Left Logical Shift Right	MOVS Rd, Rs, LSL Rs MOV Rd, Rd, LSR Rs
ASR ROR	Arithmetic Shift Right Rotate Right	MOVS Rd, Rd, ASR Rs MOVS Rd, Rd, ROR Rs
CMP NEG	Compare Negate	CMP Rd, #offset8 RSBS Rd, Rs, #0
CMN BIC	Compare Negative Bit Clear	CMN Rd, Rs BICS Rd, Rd, Rs
TST B	Test Bits Branch Unconditional	TST Rd, Rs B {label}
Bcc BL	Branch Conditional Branch and Link	B[xx] {label} BL {label}
BX MOV	Branch and Exchange Move Register/Register	n/a MOVS Rd, #offset8
MVN LDR	Move and Invert Load Word	MVNS Rd, Rs LDR Rd, [PC, #imm5]
LDRB LDRSB*	Load Byte Load Signed Byte	LDRB Rd, [Rb, #imm5] n/a
LDRH* LDRSH*	Load Halfword Load Signed Halfword	n/a n/a
LDMIA STR	Load Multiple Store Word	LDMIA Rb!, {Reg List} STR Rd, [Rb, Ro]
STRB STRH*	Store Byte Store Halfword	STRB Rd, [Rb, Ro] n/a
STMIA PUSH	Store Multiple Push Registers	STMIA Rb!, {Reg List} STMDb R13!, {Reg List}
POP SWI	Pop Registers Software Interrupt	LDMIA R13!, {Reg List} SWI value8

Thumb Instructions Umsetzung

- Umsetzung der Thumb-Befehle in die entsprechenden ARM Befehle zur Laufzeit
  - Verwendung einer Lookup-Tabelle
  - Thumb benötigt nur 3000 Transistoren (5% der Core-Fläche, optionale Makrozelle)
- Thumb Befehlsdecodierung:




```
cpsr = nzcvqifT_SVC
```

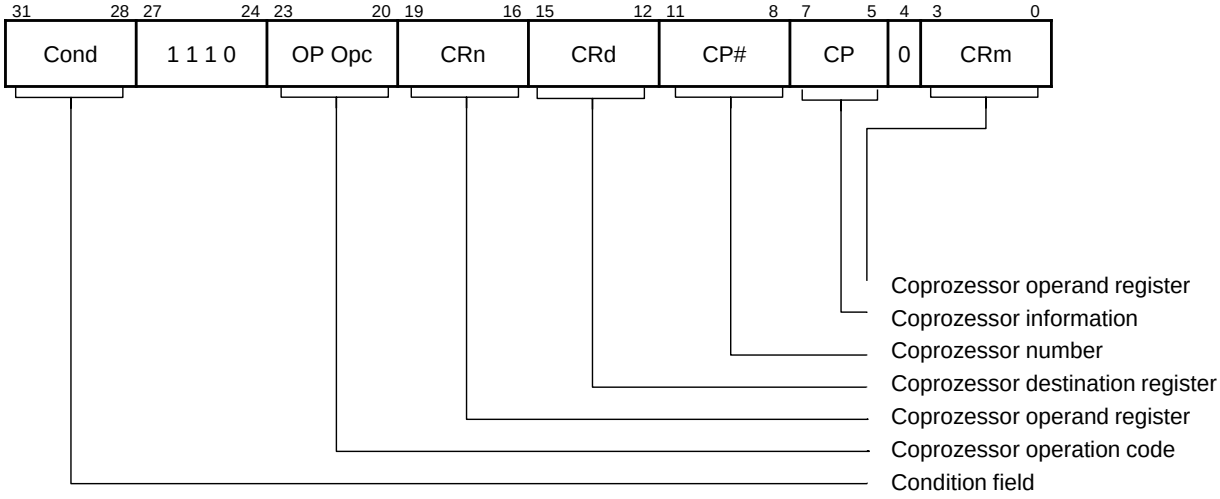

Co-Pozessor

ARM-Design ermöglicht einfache Erweiterungen durch die Koprozessor-Architektur

- Bis zu 16 Koprozessoren an einem ARM-Kern
- ARM ist für Flusssteuerung zuständig
- Lagert Berechnungen aus
- Kommunikation über Bus und Ausnahmen
- Wenn kein Koprozessor reagiert, gibt es eine undefined-Ausnahme
- Behandlung dieser Ausnahmen ermöglicht Software-Emulation
- Koprozessoren:
  - Wie ARM load/store
  - Bis zu 16 Register
  - Beliebige interne Verarbeitungsbreite

z.B FPUs (Gleitkommaeinheit), inzwischen in den Prozessor integriert,

Co-Prozessor Data Operation



ARM Prozessor Implementationen

- Unterscheiden sich je nach Lizenznehmer stark
- Lizenznehmer (u.a.):
  - Motorola
  - IBM
  - Texas Instruments
  - Nintendo
  - Atmel (AT91 Thumb)
  - Samsung
  - DEC/Intel (StrongARM, später XScale)
- Siehe auch: http://vvww.arm.com/markets

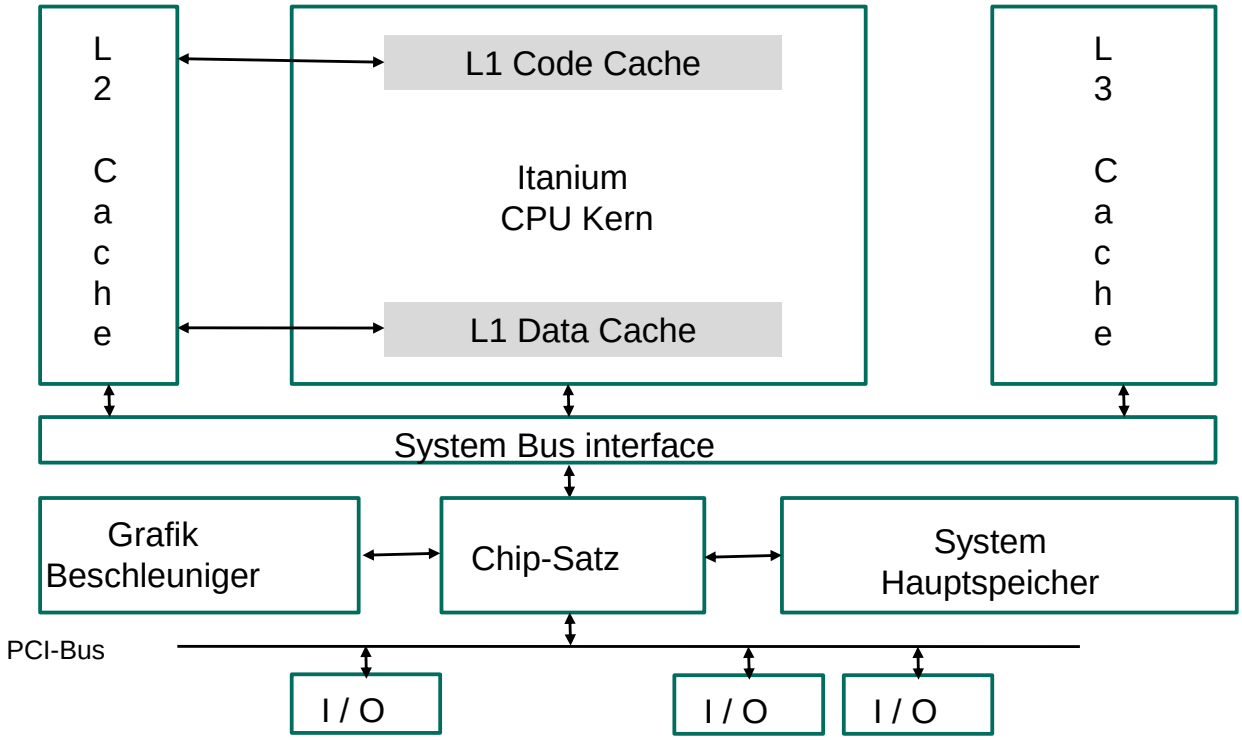
Entwicklungstools

- Abhängig von benutzter Hardware und Betriebssystem
- Allgemein:
  - ARMulator
  - SWARM
  - GNU toolchain für ARM
- Speziell:
  - Microsoft Visual Studio für Pocket PC
  - PalmOnes Entwicklungsumgebung für PalmOS
  - OpenEmbedded für linux-basierte PDAs und eingebettete Systeme

Weiteres Beispiel

Intel & HP IA64 RISC Architektur

Beispiel: Intel & HP IA64 RISC Architektur



Intel IA64 Architektur

- IA 64 Registersatz

GR 0 .. GR 127	General Register	64-Bit
GR 32 .. Gr 127	Register Stack Frame	
BR 0 .. BR 7	Branch Register	64-Bit
IP	Instruction Pointer	64-Bit
FR 0 .. FR 127	Floating Point Register	82-Bit
AR 0 .. AR 127	Application Register	64-Bit
PR 0 .. PR 63	Predicate Register	1-Bit

Intel IA64 Registersatz

- 256 application registers:
  - 128 64 bit general purpose registers (integer and multimedia):
    32 general registers (GR0 - GR31): static, available to all programs,
    rest (GR32 - GR127) stacked: available per program,
    managed by Register Stack Engine (RSE) (stack pointer (SP): Current Frame Marker, CFM)
  - 128 82 bit floating point registers (FR0 - FR127):
    first 32 registers static: available to all programs,
    rest rotating: can be renamed to accelerate loops.
- 64 predicate registers (PR0 - PR63): contain predicate test (compare) results, for conditional execution of instructions, first 16 registers static: available to all programs, rest rotating: can be renamed to accelerate loops.
- 8 branch registers (BR0 - BR7)
- 128 application registers (AR0 - AR127): special-purpose data and control registers.
- 4 Privilege Levels (PL): 0-3.
  Current Privilege Level (CPL) in PSR.cpl (Processor Status Register, PSR)
- Bi-endian memory access: controlled by UM.be bit (User Mask, UM).
- Memory mapped I/O.

Cache Architektur

- On-die L1 cache (Harvard architecture):
  - 16 kbyte instruction cache (L1I):
    4-way set-associative, 32 byte line size,
    1 cycle latency,
  - 16 kbyte data cache (L1D):
    4-way set-associative, 32 byte line size,
    write-through, no write-allocate,
    2 cycles latency for integers, FPs bypass the L1 data cache.
- On-die, unified L2 cache:
  96 kbyte,
  6-way set-associative, 64 byte line size,
  write back,
  6 cycles minimum latency for integers, 9 cycles minimum latency for FPs,
  max. 2 requests per clock (banks)
- L3 cache:
  2 or 4 Mbyte, apart in package, connected through Front Side Bus (FSB),
  4-way set-associative, 64 byte line size,
  21 cycles minimum latency for integers, 24 cycles minimum latency for FPs,
  bandwidth 16 bytes per core cycle (64 bit DDR, 128 bit bus to core),
  12 Gbyte/s max. throughput.

Prozessor Architektur (1)

- Double pipeline: 10 stage in-order, 6 instructions wide.
  Split issue dispersal: three instructions (16 bytes) per bundle
- 17 execution units:
  - 4 Integer units (ALU, Arithmetic Logic Unit),
  - 4 Multimedia units (ALU),
  - 2 Extended Precision Floating Point (FP) units: F0, F1,
    ANSI/IEEE-754,
    FMAC: Floating Point Multiply Add Calculation: multiply and add of 82 bit floating point values in one cycle (for matrix calculations),
  - 2 Single Precision FPUs,
    each executing two calculations per clock,
  - all FP units together delivering max. throughput of 6.4 GFLOPS,
  - 2 load/store units: M0, M1,
  - 3 branch units: B0, B1, B2
- 9 issue ports:
  - 2 memory: M0, M1,
  - 2 integer: I0, I1,
  - 2 FP: F0, F1,
  - 3 branch: B0, B1, B2,
  - serving the 17 execution units above.

Prozessor Architektur (2)

- Dynamic prefetch, branch prediction, speculative execution.
- Branch prediction:
  512 entry, two-level.
  Branch Target Address Cache (BTAC): 64 entry.
- IA-32 compatibility mode: IA-32 System Environment, i.e. Pentium III.
  16 bit Real Mode, 16 bit VM86, 16/32 bit Protected Mode, memory segmentation.
  Multimedia instruction sets: MMX, SSE
- Interval Time Counter (ITC): register for timing ticks.
  In 32 bit compatibility mode: Time Stamp Counter (TSC).
- Streamlined Advanced PIC (SAPIC):
  based on IA-32 APIC (Advanced Programmable Interrupt Controller),
  for Aborts, Interrupts, Faults, and Traps:
  handled by operating system: to Interrupt Vector Address (IVA) through Interrupt Vector Table (IVT),
- Interruption Status Register (ISR).
  256 interrupt vectors: 0 - 15: special, high priority,
  16 - 255: freely assignable.
- Support for Intel 8259A interrupt controllers.

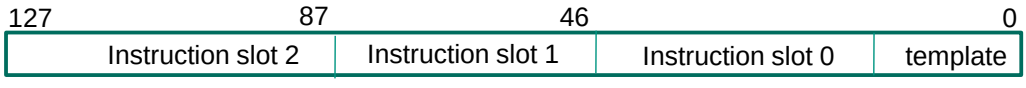
Prozessor Architektur (3)

- Virtual address space: 64 bit, no segmentation.
  Multiple Address Space (MAS): each process has its own unique Virtual Region (flat linear address space).
  8 61 bit Virtual Regions (Virtual Region Number, VRN; Region Identifier, RID), 224 Virtual Address Spaces of 261 bits.
  4 kbyte - 256 Mbyte pages (Virtual Page Number, VPN).
- Physical address space: 63 bit.
  Up to 50 bits supported in page tables.
- Write Coalescing (WC): streams of non-cachable writes can be combined into a single bus write transaction.
  WC Buffer (WCB): two-entry, 64 byte.
- Enhanced Machine Check Architecture (EMCA): parity and ECC (Error-Correcting Code) on all major address and data busses.
- 44 bit address bus.
  Physical addressing:
  - 32 bit: 0-4 Gbyte,
  - 36 bit: 4-64 Gbyte,
  - 44 bit: 64 Gbyte - 16 Tbyte.

Befehlsvorrat

- Befehlsgruppen
  - Registerstack-Befehle
  - Integer Berechnungsbefehle
  - Vergleichsbefehle und Predication
  - Speicherzugriffsbefehle
  - Verhaltenssteuerung Speicherhierarchie
  - Prefetching Mechanismen
  - Verzweigungsbefehle
  - Schleifenunterstützung

- IA64 Befehlsformat: 128 Bit lang, bündelt 3 je 41 Bit lange Befehlsslots und ein 5 Bit Template Feld (EPIC explicitly parallel inctruction computing)



Befehlsvorrat Intel IA32 Architektur

Vergleiche alte Intel Pentium CISC-Architektur

Siehe **X86 32 Bit Instruction Set Architecture (ISA)**

http://www.chiplist.com/ISA/X86_32/