6. CPU, die Zweite

Stand der Technik

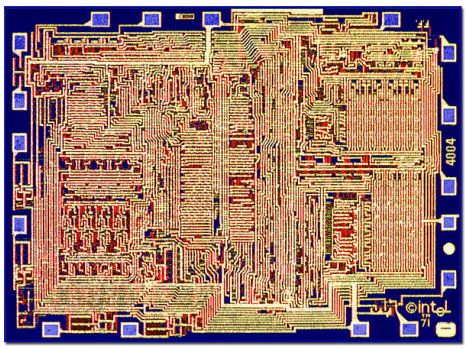
Vorlesung Rechnerstrukturen
Wintersemester 2002/03

Ausgangspunkt

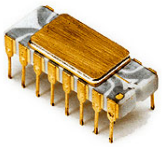
- Tanenbaum-CPU
- 3 sichtbare Register
  - PC = Program Counter
  - AC = Accumulator
  - SP = Stack Pointer
- Mikroprogrammierte Architektur
  - zwischen 8 und 12 Mikroinstruktion pro Instruktion
- Keine käuflichen Muster bekannt

Geschichte des µProzessors

- Früher: Zentraleinheit
  - aus diskreten Komponenten aufgebaute CPU
- Mikroprozessor = CPU auf einem Chip
- Intel 4004
  - 1971, 4-Bit Prozessor
- Intel 8008
  - 1972, 8-Bit Version



Intel 4004 Mikroprozessor

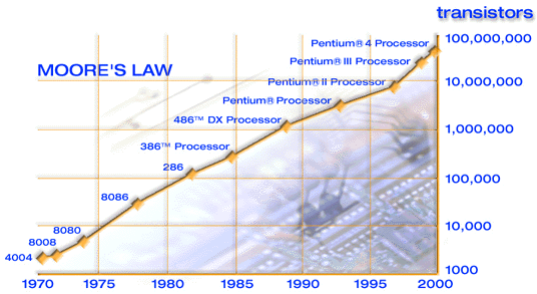


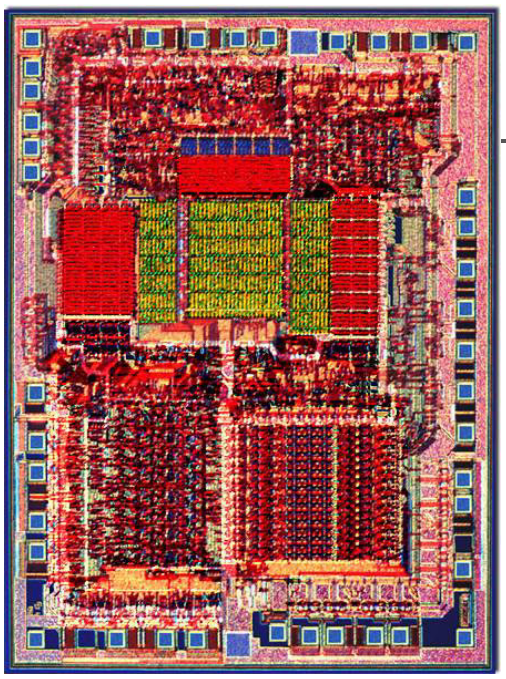
4001, 4002, 4003, 4004

- Programmierter elektronischer Taschenrechner
  - Intel traut sich zuerst integrierte Lösung zu entwicklen
  - Gelötete diskrete Lösungen waren bis dahin üblich
- Hohe und riskante Anfangsinvestition
- Bestandteile
  - 4001: 2048 Bit ROM, 4 Bit E/A-Port
  - 4002: 20x4 Bit RAM, 4 Bit Ausgangsport
  - 4003: Shift-Register (Seriell Ein, Seriell/Parallel Aus)
  - 4004: 4 Bit CPU
- siehe auch
  - http://www.intel4004.com/
  - http://www.intel.com/intel/intelis/museum/exhibit/hist_micro/hof/hof_main.htm

Intel

- 1968 –
  - Gordon Moore (Moore's Law)
  - Robert Noyce
  - später Andy Grove
- „Erfinder" des Mikroprozessors
- Buchempfehlung
  - Tim Jackson, *Inside Intel*, Andy Grove and the Rise of the World's Most Powerful Chip Company, 1998, Plume Book (deutsche Fassung im Heyne-Verlag)



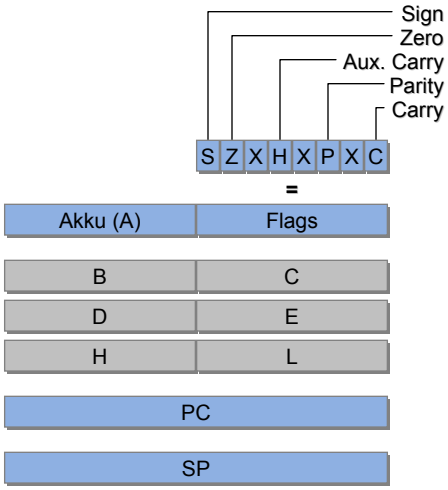


Der Intel 8085

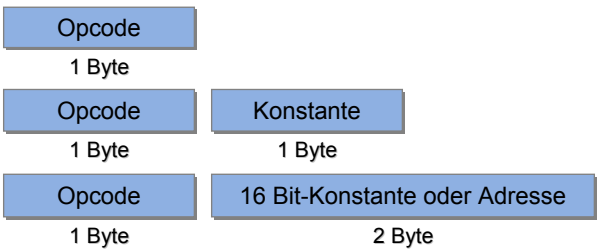
- 8-Bit CPU
  - auch 16-Bit-Register
  - 16-Bit-Adressen (64 KB Speicher)
- Einführung 1974
- max. 2 MHz Takt
- 6000 Transistoren
- Auflösung 6 µm

Sichtbare Registerstruktur

- 16 Bit Adressen
- 8 Bit-Register
  - Akkumulator
  - Flags
  - B, C, D, E, H, L
- 16 Bit-Register
  - Programmzähler (PC)
  - Stackpointer (SP)
  - PSW = Akku + Flags
  - BC, DE, HL
- Viele Spezialinstruktionen
  - Nur bestimmte Register verwendbar



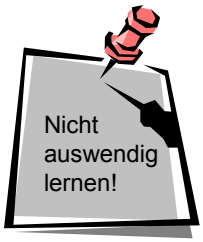
Instruktionsformate



- Little Endian
  - Reihenfolge der einzelnen Bytes bei Mehr-Byte-Größen
  - Beispiel `0x34af` ab Adresse `0x2000`
    - `0x2000: 0xaf`
    - `0x2001: 0x34`

Befehlsgruppen

- Datentransfer
  - Register-Register
  - Register-Speicher
  - Speicher-Register
  - Immediate
- Arithmetisch-logische Befehle
  - Addition mit und ohne Carry
  - Subtraktion mit und ohne Borrow
  - Inkrement und Dekrement
  - Logisch Und, Oder, Exklusiv-Oder
  - Vergleich
  - Shiften und Rotieren
- Sprungbefehle
- Unterprogrammbefehle
- Kelleroperationen
- Spezialbefehle



Datentransfer

- 8 Bit Register-Register
  - **MOV r1, r2**
    - r1 := r2, mit r1, r2 aus { A, B, C, D, E, H, L }
- 16 Bit Register-Register
  - **XCHG**
    - tmp16 := DE
DE = HL
HL = tmp16
  - **PCHL**
    - PC := HL
  - **SPHL**
    - SP := HL
- 8 Bit Register-Speicher
  - **MOV r, M**
    - r := m[HL], mit r aus { A, B, C, D, E, H, L }
  - **LDAX rr**
    - A := m[rr], mit rr aus { BC, DE, HL }
  - **STA adr16**
    - m[adr16] := A
- 16 Bit Register-Speicher
  - **LHLD adr16**
    - L := m[adr16]
H := m[adr16+1]

Datentransfer (2)

- 8 Bit Speicher-Register
  - **MOV M, r**
    - m[HL] := r, mit r aus { A, B, C, D, E, H, L }
  - **STAX rr**
    - m[rr] := A, mit rr aus { BC, DE, HL }
  - **STA adr16**
    - m[adr16] := A
- 16 Bit Speicher-Register
  - **SHLD adr16**
    - m[adr16] := L
      m[adr16+1] := H
- 8 Bit Immediate
  - **MVI r, byte**
    - r := byte, mit r aus { A, B, C, D, E, H, L, M }
- 16 Bit Immediate
  - **LXI rr, dbyte**
    - rr := dbyte, mit r aus { BC, DE, HL, SP }

Arithmetik

- 8 Bit Addition
  - **ADD r**
    - A := A + r, r aus { A, B, C, D, E, H, L, M }
    - Flags: C, Z, S, P, AC
  - **ADC r**
    - A := A + r + Carry
    - Flags: C, Z, S, P, AC
  - **ADI byte**
    - A := A + byte
    - Flags: C, Z, S, P, AC
  - **ACI byte**
    - A := A + byte + Carry
    - Flags: C, Z, S, P, AC
- Analog Subtraktion
  - **SUB**, **SBB**, **SUI**, **SBI**
- 16 Bit Addition
  - **DAD rr**
    - HL := HL + rr, rr aus { BC, DE, HL, SP }
    - Flags: C
- 8 Bit Inkrement / Dekrement
  - **INR r**
    - r := r+1, r aus { A, B, C, D, E, H, L, M }
    - Flags: Z, S, P, AC
  - **DCR r**
    - r := r-1, r aus { A, B, C, D, E, H, L, M }
    - Flags: Z, S, P, AC

Arithmetik (2)

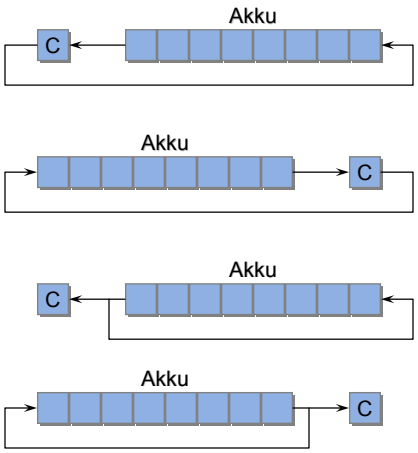
- 16 Bit Inkrement / Dekrement
  - **INX rr**
    - rr := rr+1, rr aus { BC, DE, HL, SP }
  - **DCX rr**
    - rr := rr-1, rr aus
    - { BC, DE, HL, SP }
- Vergleichen
  - **CMP r**
    - tmp8 := A - r
    - Flags: C, Z, S, P, AC
  - **CPI byte**
    - tmp8 := A - byte
    - Flags: C, Z, S, P, AC
- Carry setzen
  - **STC**
    - Flag C := 1
- Carry invertieren
  - **CMC**
    - Flag C := inv(Flag C)
- BCD-Korrektur
  - **DAA**
- Akku invertieren
  - **CMA**
    - A := inv(A)

Logik

- 8 Bit Logisches UND
  - **ANA r**
    - A := A AND r, r aus { A, B, C, D, E, H, L, M }
    - Flags: C:=0, Z, S, P, AC
  - **ANI byte**
    - A := A AND byte
    - Flags: C:=0, Z, S, P, AC
- Analog Logisches ODER
  - **ORA r**
  - **ORI byte**
- Analog Logisches Exklusiv-ODER
  - **XRA r**
  - **XRI byte**

Rotieren

- **RAL**
  - tmp1 := Carry
    Carry := A7
    A7 := A6, ..., A1 := A0
    A0 := tmp1
- **RAR**
  - tmp1 := Carry
    Carry := A0
    A0 := A1, ..., A6 := A7
    A7 := tmp1
- **RLC**
  - Carry := A7
    A7 := A6, ..., A1 := A0
    A0 := Carry
- **RRC**
  - Carry := A0
    A0 := A1, ..., A6 := A7
    A7 := Carry



Sprünge und Unterprogramme

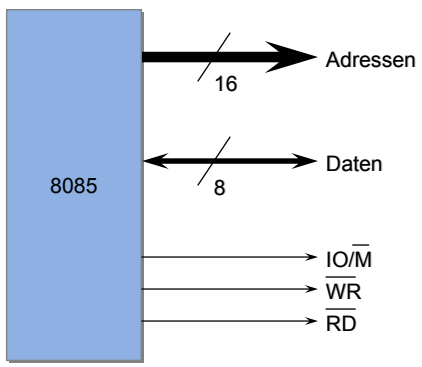
- Unbedingter Sprung
  - **JMP adr16**
- Bedingter Sprung
  - **Jcc adr16**
  - cc =
    - NZ: Not Zero
    - Z: Zero
    - NC: No Carry
    - C: Carry
    - PO: Parity Odd
    - PE: Parity Even
    - P: Positive
    - M: Minus (Negative)
- Indirekter Sprung
  - **PCHL**
    - PC := HL
- Unterprogrammsprung
  - **CALL adr16**
  - **Ccc adr16**
    - SP := SP-1
      m[SP] := PC_{MSB}
      SP := SP-1
      m[SP] := PC_{LSB}
      PC := adr16
- Unterprogrammrücksprung
  - **RET**
  - **Rcc**
    - PC_{LSB} := m[SP]
      SP := SP+1
      PC_{MSB} := m[SP]
      SP := SP+1

Kellerbefehle

- Push
  - **PUSH rr**
    - rr aus { BC, DE, HL, PSW }
    - SP := SP-1, m[SP] := rr_{MSB}, SP := SP-1, m[SP] := rr_{LSB}
- Pop
  - **POP rr**
    - rr aus { BC, DE, HL, PSW }
    - rr_{LSB} := m[SP], SP := SP+1, rr_{MSB} := m[SP], SP := SP+1
- Spezialbefehle
  - **XTHL**
    - tmp8 := L, L := m[SP], m[SP] := tmp8
      tmp8 := H, H := m[SP+1], m[SP+1] := tmp8
  - **SPHL**
    - SP := HL // Der Vollständigkeit halber

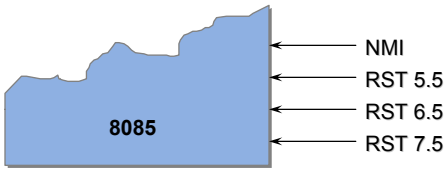
Ein- und Ausgabe

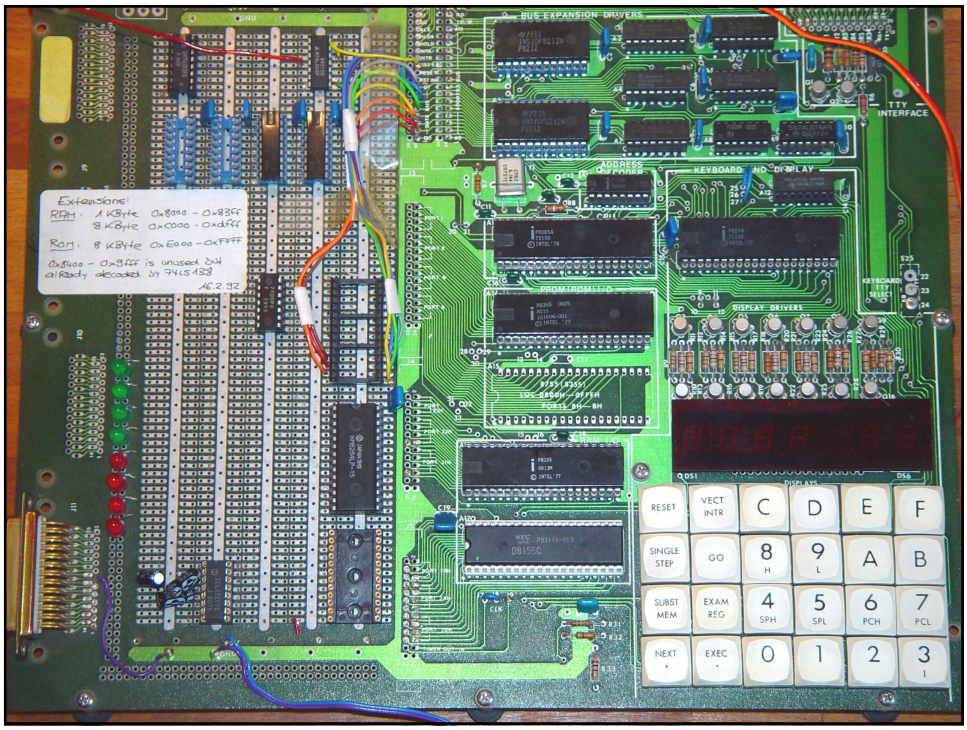
- 256 E/A-Adressen
- Ausgabe
  - **OUT adr8**
    - EA[adr8] := A
- Eingabe
  - **IN adr8**
    - A := EA[adr8]

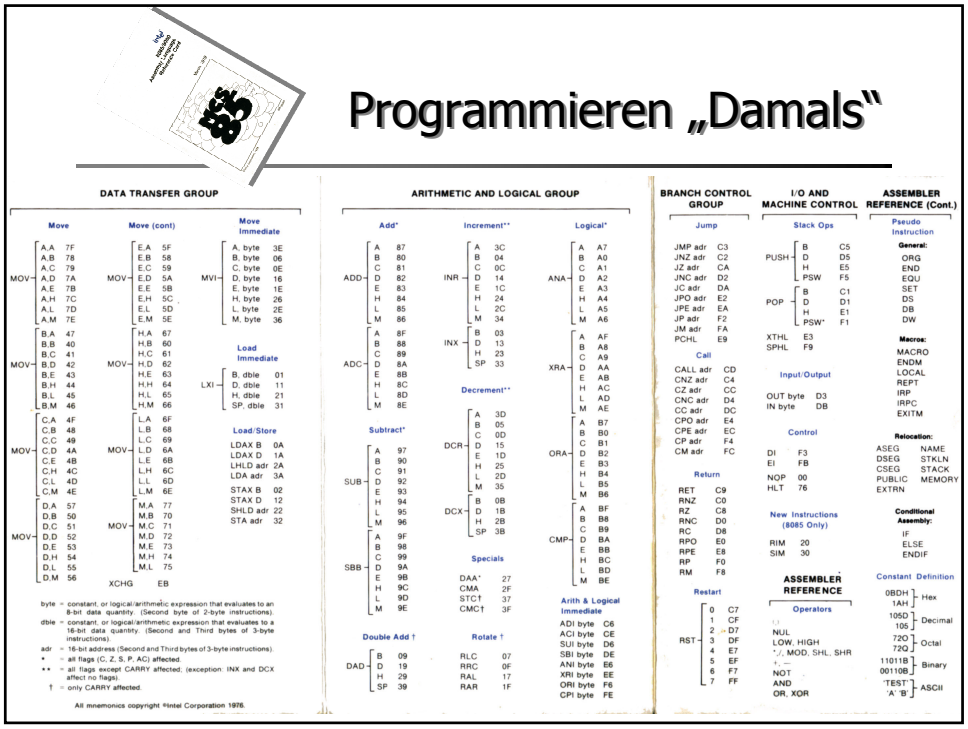


Interrupt-Behandlung

- Software-Interrupt
  - **RST x**
    - SP := SP-1
m[SP] := PC_{MSB}
SP := SP-1
m[SP] := PC_{LSB}
PC := m[x*8]
- Hardware-Interrupts
  - 4 Stück
  - RST 4.5 (NMI), 5.5, 6.5, 7.5
- Interruptmaske
  - **RIM**
    - A := IM
  - **SIM**
    - IM := A
- Interrupts zulassen
  - **EI**
- Interrupts sperren
  - **DI**
- Misc
  - **HLT**
  - **NOP**







Prozessoren der Vergangenheit

- Keine detaillierte Untersuchung
- Vergleich charakteristischer Eigenschaften
  - Maximale Taktfrequenz (zum Einführungszeitpunkt)
  - Anzahl Transistoren
  - Instruktionen pro Takt
  - Cachegrößen (eventuell Summe)
  - Tiefe der Instruktionspipeline (Integer und Floating Point)

Intel: 70er Jahre

	4004	8008	8080	8086	8088
Introduced	11/15/71	4/1/72	4/1/74	6/8/78	6/1/79
Clock Speeds	108KHz	200KHz	2MHz	5MHz, 8MHz, 10MHz	5MHz, 8MHz
Bus Width	4 bits	8 bits	8 bits	16 bits	8 bits
Number of Transistors	2,300 (10 microns)	3,500 (10 microns)	6,000 (6 microns)	29,000 (3 microns)	29,000 (3 microns)
Addressable Memory	640 bytes	16 KBytes	64 KBytes	1 MB	1 MB
Virtual Memory	--	--	--	--	--
Brief Description	First microcomputer chip, Arithmetic manipulation	Data/character manipulation	10X the performance of the 8008	10X the performance of the 8080	Identical to 8086 except for its 8-bit external bus

Referenz: www.intel.com

Intel: 80er Jahre

	80286	Intel386™ DX Microprocessor	Intel386™ SX Microprocessor	Intel486™ DX CPU Microprocessor
Introduced	2/1/82	10/17/85	6/16/88	4/10/89
Clock Speeds	6MHz, 8MHz, 10MHz, 12.5MHz	16MHz, 20MHz, 25MHz, 33MHz	16MHz, 20MHz, 25MHz, 33MHz	25MHz, 33MHz, 50MHz
Bus Width	16 bits	32 bits	16 bits	32 bits
Number of Transistors	134,000 (1.5 microns)	275,000 (1 micron)	275,000 (1 micron)	1.2 million (1 micron) (.8 micron with 50MHz)
Addressable Memory	16 megabytes	4 gigabytes	16 megabytes	4 gigabytes
Virtual Memory	1 gigabyte	64 terabytes	64 terabytes	64 terabytes
Brief Description	3-6X the performance of the 8086	First X86 chip to handle 32-bit data sets	16-bit address bus enabled low-cost 32-bit processing	Level 1 cache on chip

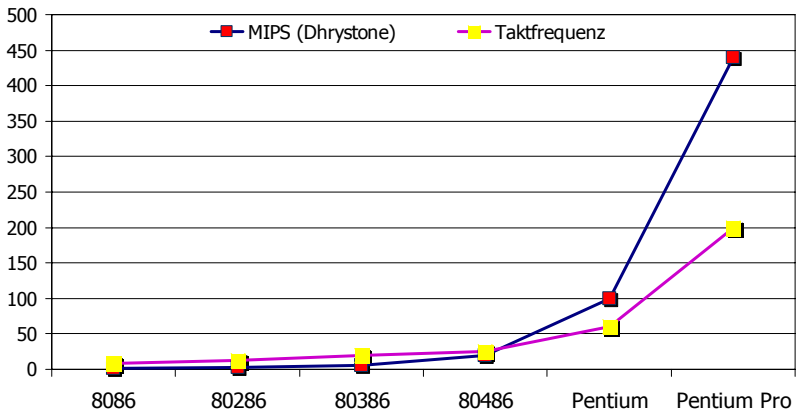
Referenz: www.intel.com

Intel: 90er Jahre

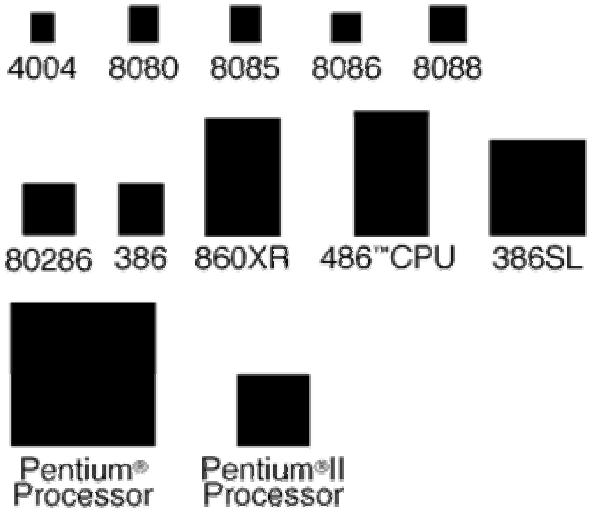
	Intel486™ SX Microprocessor	Pentium® Processor	Pentium® Pro Processor	Pentium® II Processor
Introduced	4/22/91	3/22/93	11/01/95	5/07/97
Clock Speeds	16MHz, 20MHz, 25MHz, 33MHz	60MHz,66MHz	150MHz, 166MHz, 180MHz, 200MHz	200MHz, 233MHz, 266MHz, 300MHz
Bus Width	32 bits	64 bits	64 bits	64 bits
Number of Transistors	1.185 million (1 micron)	3.1 million (.8 micron)	5.5 million (0.35 micron)	7.5 million (0.35 micron)
Addressable Memory	4 gigabytes	4 gigabytes	64 gigabytes	64 gigabytes
Virtual Memory	64 terabytes	64 terabytes	64 terabytes	64 terabytes
Brief Description	Identical in design to Intel486™ DX but without math coprocessor	Superscalar architecture brought 5X the performance of the 33-MHz Intel486™ DX processor	Dynamic execution architecture drives high-performing processor	Dual independent bus, dynamic execution, Intel MMX™ technology
Other Processor Family Members	Quick Reference Guide	Quick Reference Guide	Quick Reference Guide	Quick Reference Guide

Referenz: www.intel.com

Instruktionen pro Takt



Die-Größen: Intel

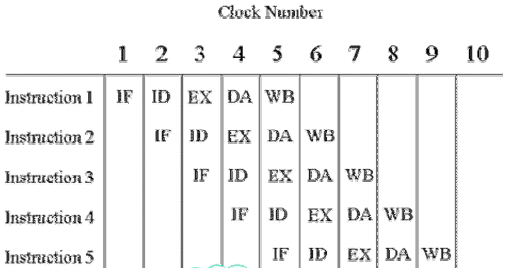


Schneller, schneller, schneller …

- Höhere Leistung ist oberstes Gebot
- Technische Verbesserungen
  - Kupfer statt Aluminium, eventuell mal GaAs
  - Höhere Integration (90nm und besser)
  - Mehrere Prozessoren auf einen Chip
  - Ganze Rechner auf einem Chip
  - …
- Algorithmische Verbesserungen
  - Instruction Pipelining zusammen mit Branch Prediction
  - Ausführung von Instruktionen „Out of Order"

Instruction Pipelining

- Taktfrequenz erhöhen
- Pipeline Hazards führen zu einem Stall
  - Data Hazard (benötigte Daten nicht verfügbar)
  - Control Hazard (bedingter Sprung etc.)
  - Structural Hazard (Engpäße in der CPU)
- Vermeidung von Stalls bei Control Hazards
  - Branch Prediction



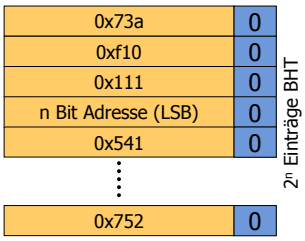
Branch Prediction

- Sprungarten
  - Bedingt Vorwärts
  - Bedingt Rückwärts
  - Unbedingt

- Verteilung
  - Vorwärts : Rückwärts ca. 4:1 (Bedingt und Unbedingt)
  - ca. 60% der Vorwärtssprünge werden genommen
  - ca. 85% der Rückwärtssprünge werden genommen
- Erster Lösungsansatz: Static Predictor
  - Vergleich aktuelle Adresse und Zieladresse

Dynamische Sprungvorhersage

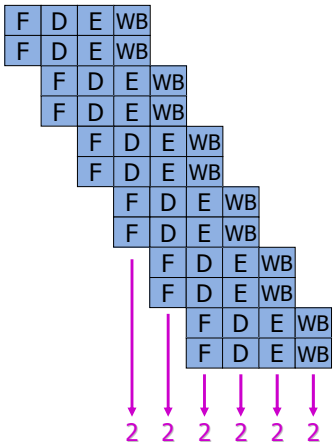
- Spezieller Cache
  - Index: Adresse
  - Inhalt: 1 Bit (Gesprungen oder Nicht)
- Branch History Buffer (BHB)
  Branch History Table (BHT)
- 2 Bit sind besser (Gesättigter Zähler)
  - Inkrement / Dekrement
  - 1 Bit bei Mehrfachdurchläufen einer Schleife immer am Anfang und am Ende falsch (z.B. geschachtelte Schleifen)
  - mehr als 2 Bit nicht mehr sinnvoll
  - Performanz: 82% bis 99% bei Tabelle mit 4096 Einträgen



Branch Target Buffer

- Decode-Stufe erkennt eigentlich erst bedingte Sprünge
  - mit nächstem Takt muß aber schon die geeignete nächste Instruktion geladen werden
- CPU speichert Sprungziele (BTB)
  - Wenn PC im BTB dann Sprungvorhersage einsetzen
  - sonst normal weiterarbeiten (Instruction Fetch ab PC)

Superskalarität



- Voraussetzung
  - Mehrere Funktionsbausteine vorhanden
  - Mehrere Instruktionen dekodiert und ausführbar (issue)
- Überlappende Nutzung durch Pipelining
- Beispiel: 2 Addierwerke
  - 2 Instruktionen pro Takt
- Gängig
  - mind. 2 Integer-Pipelines
  - Floating-Point
  - ggf. Multimedia-Pipeline

"Out of Order" Execution

- Inorder: Sequentielle Ausführungsreihenfolge
- Out of Order
  - Internes Umordnen der Instruktionen wenn möglich
  - Bessere Auslastung der Pipeline
  - Ideal zusammen mit Superskalarität
- Problematik: Datenabhängigkeiten zwischen Istruktionen
- Fälle
  - RAR: Read after Read
  - RAW: Read after Write
  - WAR: Write after Read
  - WAW: Write after Write

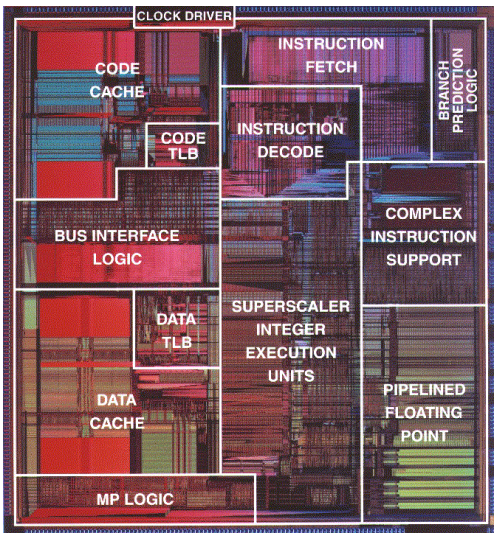
„Out of Order" Realisierung

- Scoreboards
  - Zählen wie of Register X referenziert wird
  - Merken ob Register X Ausgabeziel einer Instruktion ist
  - Anzahl vorhandener und belegter Funktionseinheiten
- Register-Umbenennung (Register Renaming)
  - „Unsichtbare" Zusatzregister
  - Auflösen der Zugriffskonflikte
- Freigabe der Ergebnisse (Commit, Retire)
  - entsprechend der ursprünglichen Instruktionsreihenfolge

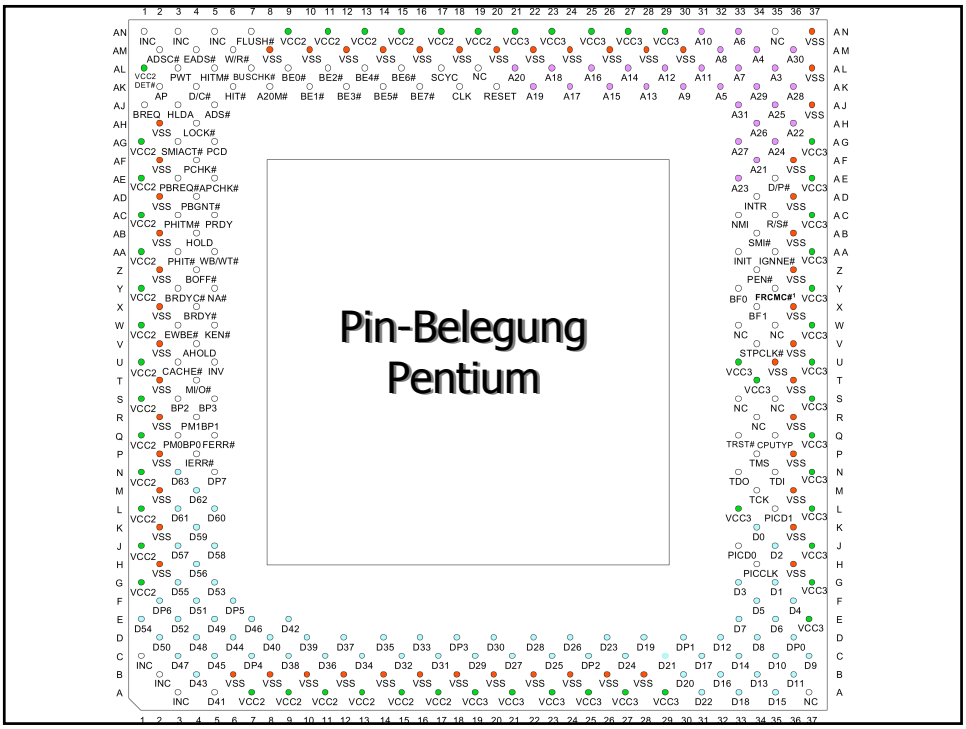
Intel Pentium

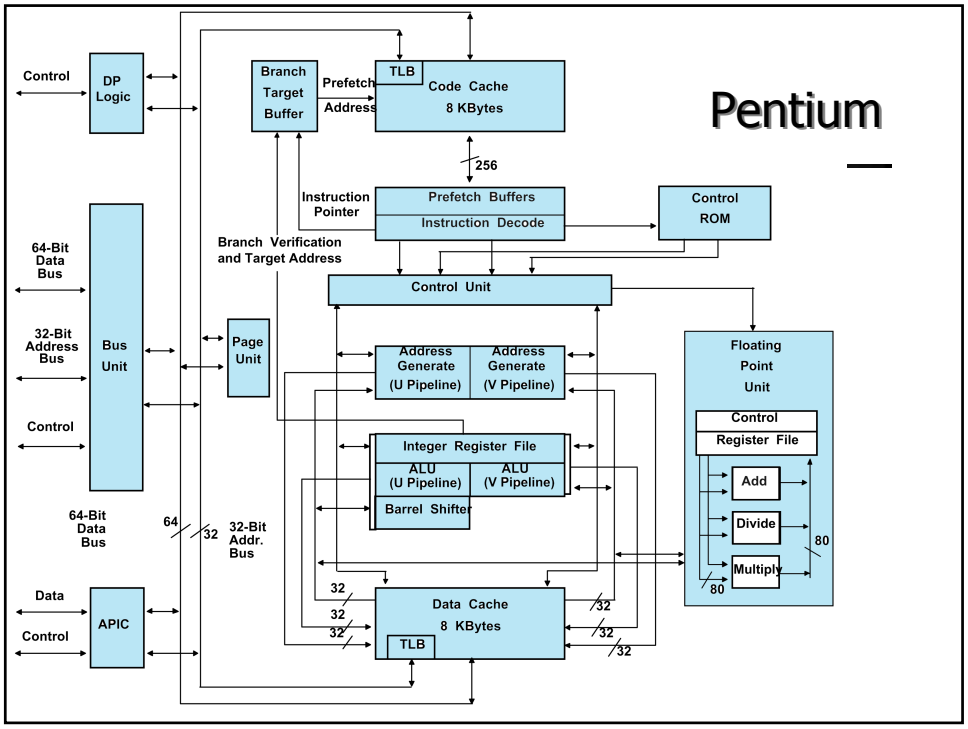
32 Bit

Pentium-Familie

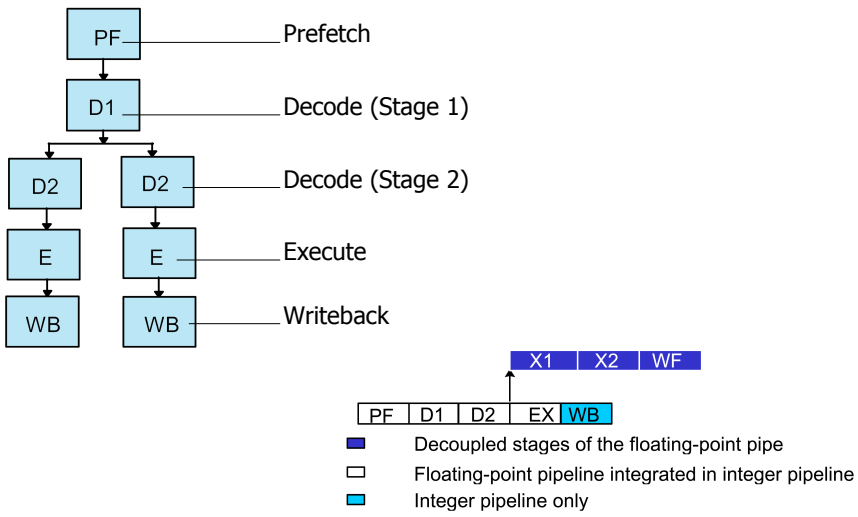


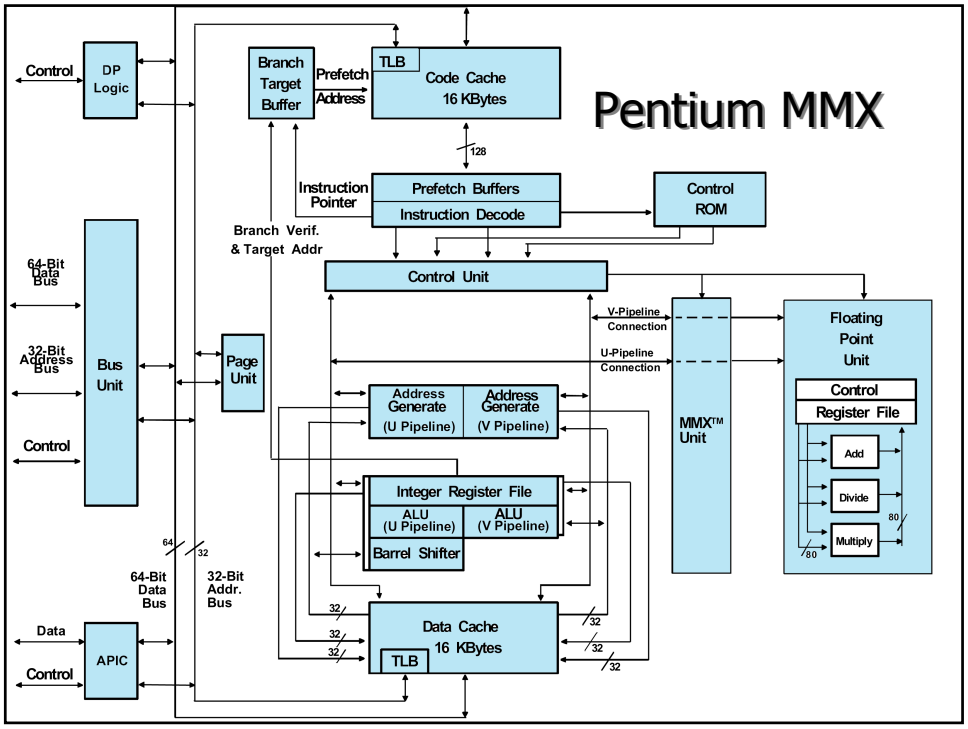
- Diverse Unterarten
  - Pentium bis Pentium 4
  - Pentium Pro
  - Xeon
  - Celeron

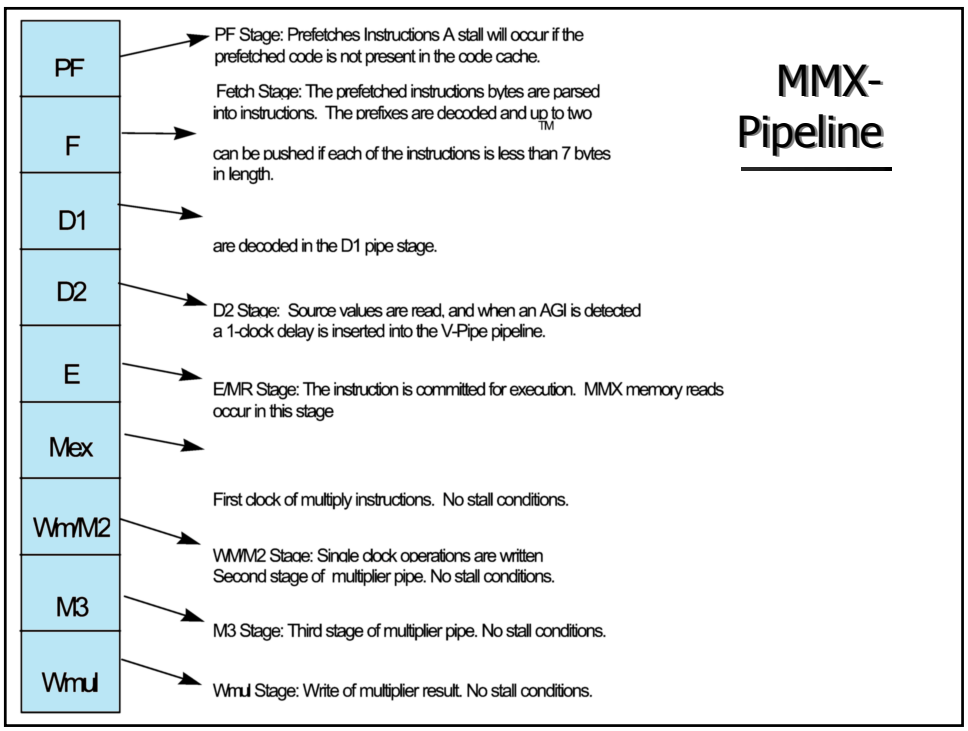


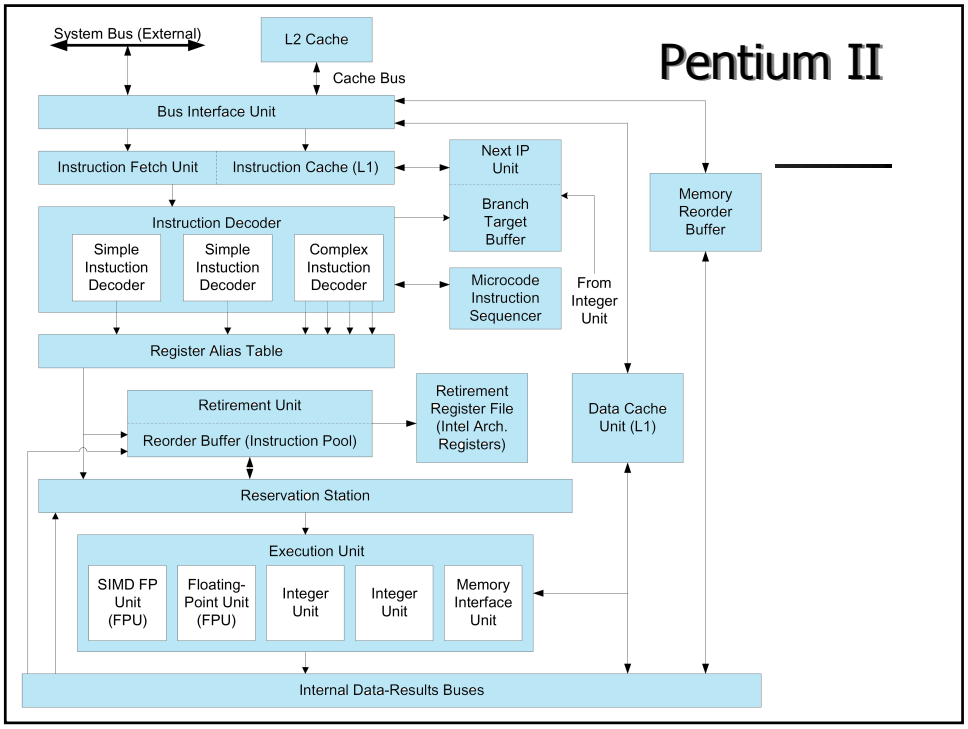


Integer- und FP-Pipeline



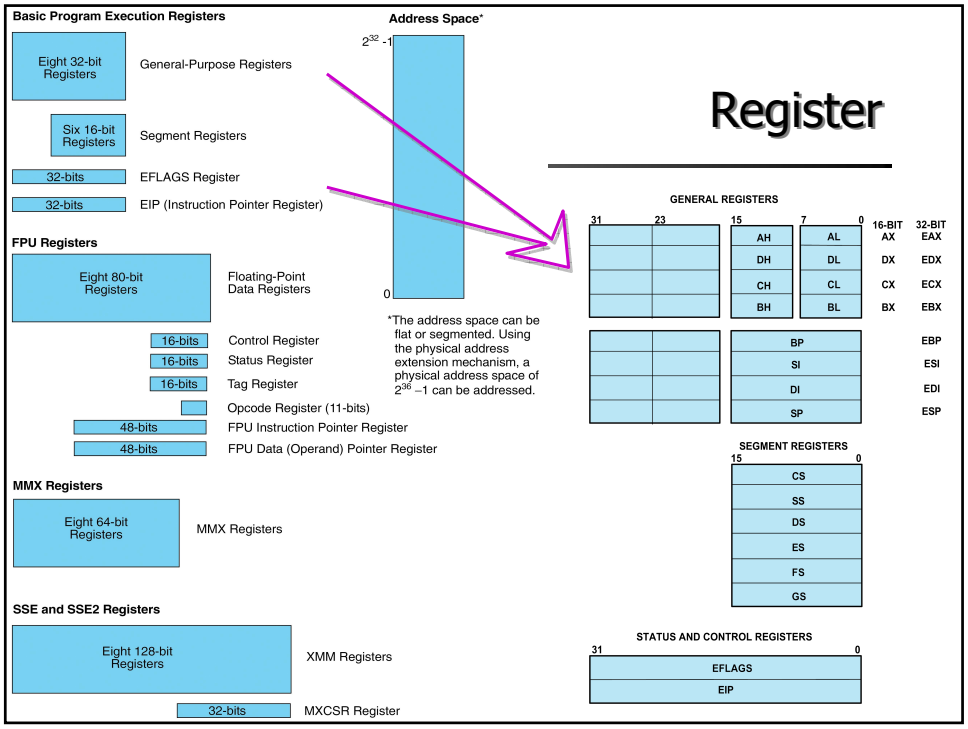

Pentium MMX
Control
DP Logic
Branch Target Buffer
Prefetch Address
TLB
Code Cache 16 KBytes
128
Instruction Pointer
Prefetch Buffers
Instruction Decode
Control ROM
Branch Verif. & Target Addr
Control Unit
64-Bit Data Bus
32-Bit Address Bus
Control
Bus Unit
Page Unit
V-Pipeline Connection
U-Pipeline Connection
Address Generate (U Pipeline)
Address Generate (V Pipeline)
MMX™ Unit
Floating Point Unit
Control
Register File
Add
Divide
Multiply
80
Integer Register File
ALU (U Pipeline)
ALU (V Pipeline)
Barrel Shifter
64
32
64-Bit Data Bus
32-Bit Addr. Bus
Data Cache 16 KBytes
TLB
Data
Control
APIC

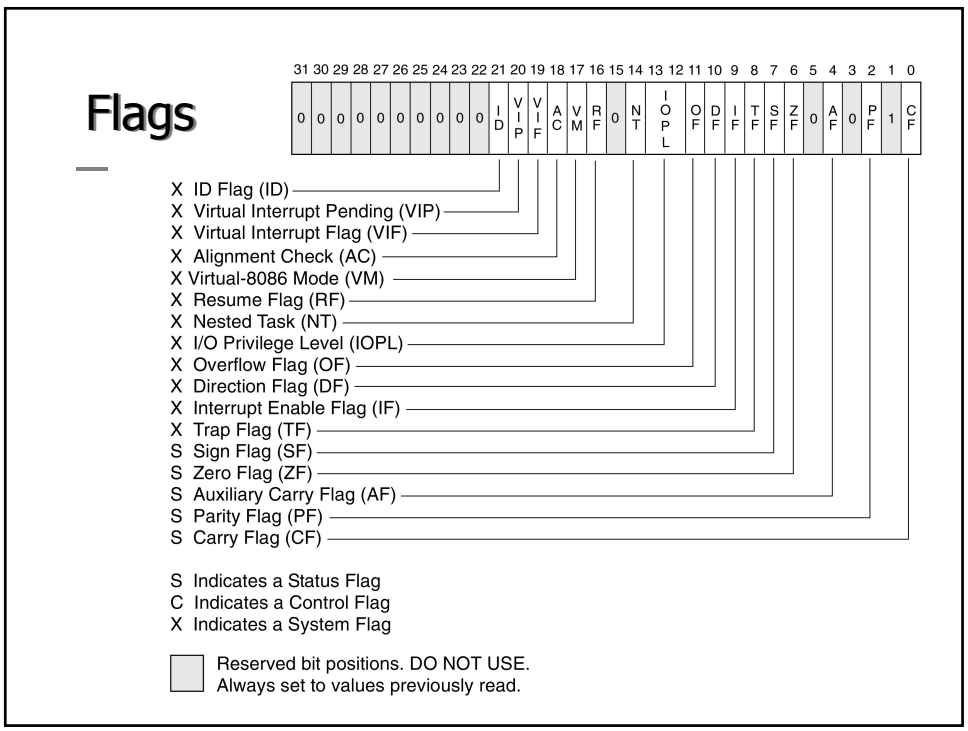

MMX-Pipeline
PF
F
D1
D2
E
Mex
Wm/M2
M3
Wmul
PF Stage: Prefetches Instructions A stall will occur if the prefetched code is not present in the code cache.
Fetch Stage: The prefetched instructions bytes are parsed into instructions. The prefixes are decoded and up to two
TM
can be pushed if each of the instructions is less than 7 bytes in length.
are decoded in the D1 pipe stage.
D2 Stage: Source values are read, and when an AGI is detected a 1-clock delay is inserted into the V-Pipe pipeline.
E/MR Stage: The instruction is committed for execution. MMX memory reads occur in this stage
First clock of multiply instructions. No stall conditions.
WM/M2 Stage: Single clock operations are written Second stage of multiplier pipe. No stall conditions.
M3 Stage: Third stage of multiplier pipe. No stall conditions.
Wmul Stage: Write of multiplier result. No stall conditions.

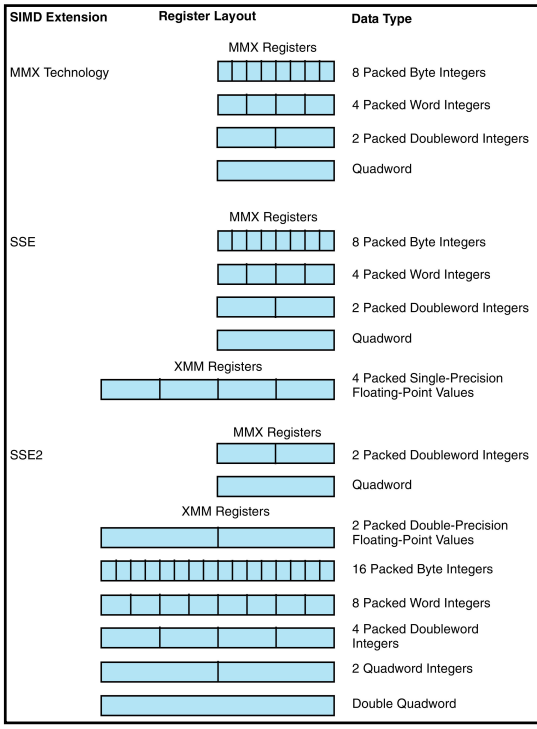


IA32: Befehlssatz

- Viele Spezialregister
- Komplexer Befehlssatz
- Viele Adressierungsarten


Basic Program Execution Registers
Eight 32-bit Registers
General-Purpose Registers
Six 16-bit Registers
Segment Registers
32-bits
EFLAGS Register
32-bits
EIP (Instruction Pointer Register)
FPU Registers
Eight 80-bit Registers
Floating-Point Data Registers
16-bits
Control Register
16-bits
Status Register
16-bits
Tag Register
Opcode Register (11-bits)
48-bits
FPU Instruction Pointer Register
48-bits
FPU Data (Operand) Pointer Register
MMX Registers
Eight 64-bit Registers
MMX Registers
SSE and SSE2 Registers
Eight 128-bit Registers
XMM Registers
32-bits
MXCSR Register
Address Space*
2^32 -1
0
*The address space can be flat or segmented. Using the physical address extension mechanism, a physical address space of 2^36 –1 can be addressed.
Register
GENERAL REGISTERS
31
23
15
7
0
16-BIT
32-BIT
AH
AL
AX
EAX
DH
DL
DX
EDX
CH
CL
CX
ECX
BH
BL
BX
EBX
BP
EBP
SI
ESI
DI
EDI
SP
ESP
SEGMENT REGISTERS
15
0
CS
SS
DS
ES
FS
GS
STATUS AND CONTROL REGISTERS
31
0
EFLAGS
EIP


Flags
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0 0 0 ID VIP VIF AC VM RF 0 NT IOPL OF DF IF TF SF ZF 0 AF 0 PF 1 CF
X ID Flag (ID)
X Virtual Interrupt Pending (VIP)
X Virtual Interrupt Flag (VIF)
X Alignment Check (AC)
X Virtual-8086 Mode (VM)
X Resume Flag (RF)
X Nested Task (NT)
X I/O Privilege Level (IOPL)
X Overflow Flag (OF)
X Direction Flag (DF)
X Interrupt Enable Flag (IF)
X Trap Flag (TF)
S Sign Flag (SF)
S Zero Flag (ZF)
S Auxiliary Carry Flag (AF)
S Parity Flag (PF)
S Carry Flag (CF)
S Indicates a Status Flag
C Indicates a Control Flag
X Indicates a System Flag
Reserved bit positions. DO NOT USE.
Always set to values previously read.



SIMD Extensions

- SIMD = Single Instruction Multiple Data
- Einsatzgebiete
  - Graphikverarbeitung
  - Multimedia (Ton, Bild)
- Viele spezielle Instruktionen
  - Gesättigte Addition
  - ...
- Leistungsgewinn
  - Gar nichts ...
  - ... Sehr viel

Table 2-1. 16-Bit Addressing Forms with the ModR/M Byte

r8(/r) r16(/r) r32(/r) mm(/r) xmm(/r) /digit (Opcode) REG =			AL AX EAX MM0 XMM0 0 000	CL CX ECX MM1 XMM1 1 001	DL DX EDX MM2 XMM2 2 010	BL BX EBX MM3 XMM3 3 011	AH SP ESP MM4 XMM4 4 100	CH BP[1] EBP MM5 XMM5 5 101	DH SI ESI MM6 XMM6 6 110	BH DI EDI MM7 XMM7 7 111
Effective Address	**Mod**	**R/M**	**Value of ModR/M Byte (in Hexadecimal)**							
[BX+SI]	00	000	00	08	10	18	20	28	30	38
[BX+DI]		001	01	09	11	19	21	29	31	39
[BP+SI]		010	02	0A	12	1A	22	2A	32	3A
[BP+DI]		011	03	0B	13	1B	23	2B	33	3B
[SI]		100	04	0C	14	1C	24	2C	34	3C
[DI]		101	05	0D	15	1D	25	2D	35	3D
disp16[2]		110	06	0E	16	1E	26	2E	36	3E
[BX]		111	07	0F	17	1F	27	2F	37	3F
[BX+SI]+disp8[3]	01	000	40	48	50	58	60	68	70	78
[BX+DI]+disp8		001	41	49	51	59	61	69	71	79
[BP+SI]+disp8		010	42	4A	52	5A	62	6A	72	7A
[BP+DI]+disp8		011	43	4B	53	5B	63	6B	73	7B
[SI]+disp8		100	44	4C	54	5C	64	6C	74	7C
[DI]+disp8		101	45	4D	55	5D	65	6D	75	7D
[BP]+disp8		110	46	4E	56	5E	66	6E	76	7E
[BX]+disp8		111	47	4F	57	5F	67	6F	77	7F
[BX+SI]+disp16	10	000	80	88	90	98	A0	A8	B0	B8
[BX+DI]+disp16		001	81	89	91	99	A1	A9	B1	B9
[BP+SI]+disp16		010	82	8A	92	9A	A2	AA	B2	BA
[BP+DI]+disp16		011	83	8B	93	9B	A3	AB	B3	BB
[SI]+disp16		100	84	8C	94	9C	A4	AC	B4	BC
[DI]+disp16		101	85	8D	95	9D	A5	AD	B5	BD
[BP]+disp16		110	86	8E	96	9E	A6	AE	B6	BE
[BX]+disp16		111	87	8F	97	9F	A7	AF	B7	BF
EAX/AX/AL/MM0/XMM0	11	000	C0	C8	D0	D8	E0	E8	F0	F8
ECX/CX/CL/MM1/XMM1		001	C1	C9	D1	D9	EQ	E9	F1	F9
EDX/DX/DL/MM2/XMM2		010	C2	CA	D2	DA	E2	EA	F2	FA
EBX/BX/BL/MM3/XMM3		011	C3	CB	D3	DB	E3	EB	F3	FB
ESP/SP/AHMM4/XMM4		100	C4	CC	D4	DC	E4	EC	F4	FC
EBP/BP/CH/MM5/XMM5		101	C5	CD	D5	DD	E5	ED	F5	FD
ESI/SI/DH/MM6/XMM6		110	C6	CE	D6	DE	E6	EE	F6	FE
EDI/DI/BH/MM7/XMM7		111	C7	CF	D7	DF	E7	EF	F7	FF

Adressierungsarten (1)

Table 2-2. 32-Bit Addressing Forms with the ModR/M Byte

r8(/r)			AL	CL	DL	BL	AH	CH	DH	BH
r16(/r)			AX	CX	DX	BX	SP	BP	SI	DI
r32(/r)			EAX	ECX	EDX	EBX	ESP	EBP	ESI	EDI
mm(/r)			MM0	MM1	MM2	MM3	MM4	MM5	MM6	MM7
xmm(/r)			XMM0	XMM1	XMM2	XMM3	XMM4	XMM5	XMM6	XMM7
/digit (Opcode)			0	1	2	3	4	5	6	7
REG =			000	001	010	011	100	101	110	111
Effective Address	**Mod**	**R/M**	**Value of ModR/M Byte (in Hexadecimal)**							
[EAX]	00	000	00	08	10	18	20	28	30	38
[ECX]		001	01	09	11	19	21	29	31	39
[EDX]		010	02	0A	12	1A	22	2A	32	3A
[EBX]		011	03	0B	13	1B	23	2B	33	3B
[--][--][1]		100	04	0C	14	1C	24	2C	34	3C
disp32[2]		101	05	0D	15	1D	25	2D	35	3D
[ESI]		110	06	0E	16	1E	26	2E	36	3E
[EDI]		111	07	0F	17	1F	27	2F	37	3F
[EAX]+disp8[3]	01	000	40	48	50	58	60	68	70	78
[ECX]+disp8		001	41	49	51	59	61	69	71	79
[EDX]+disp8		010	42	4A	52	5A	62	6A	72	7A
[EBX]+disp8		011	43	4B	53	5B	63	6B	73	7B
[--][--]+disp8		100	44	4C	54	5C	64	6C	74	7C
[EBP]+disp8		101	45	4D	55	5D	65	6D	75	7D
[ESI]+disp8		110	46	4E	56	5E	66	6E	76	7E
[EDI]+disp8		111	47	4F	57	5F	67	6F	77	7F
[EAX]+disp32	10	000	80	88	90	98	A0	A8	B0	B8
[ECX]+disp32		001	81	89	91	99	A1	A9	B1	B9
[EDX]+disp32		010	82	8A	92	9A	A2	AA	B2	BA
[EBX]+disp32		011	83	8B	93	9B	A3	AB	B3	BB
[--][--]+disp32		100	84	8C	94	9C	A4	AC	B4	BC
[EBP]+disp32		101	85	8D	95	9D	A5	AD	B5	BD
[ESI]+disp32		110	86	8E	96	9E	A6	AE	B6	BE
[EDI]+disp32		111	87	8F	97	9F	A7	AF	B7	BF
EAX/AX/AL/MM0/XMM0	11	000	C0	C8	D0	D8	E0	E8	F0	F8
ECX/CX/CL/MM/XMM1		001	C1	C9	D1	D9	E1	E9	F1	F9
EDX/DX/DL/MM2/XMM2		010	C2	CA	D2	DA	E2	EA	F2	FA
EBX/BX/BL/MM3/XMM3		011	C3	CB	D3	DB	E3	EB	F3	FB
ESP/SP/AH/MM4/XMM4		100	C4	CC	D4	DC	E4	EC	F4	FC
EBP/BP/CH/MM5/XMM5		101	C5	CD	D5	DD	E5	ED	F5	FD
ESI/SI/DH/MM6/XMM6		110	C6	CE	D6	DE	E6	EE	F6	FE
EDI/DI/BH/MM7/XMM7		111	C7	CF	D7	DF	E7	EF	F7	FF

Adressierungsarten (2)

Table 2-3. 32-Bit Addressing Forms with the SIB Byte

r32			EAX	ECX	EDX	EBX	ESP	[*]	ESI	EDI
Base =			0	1	2	3	4	5	6	7
Base =			000	001	010	011	100	101	110	111
Scaled Index	**SS**	**Index**	**Value of SIB Byte (in Hexadecimal)**							
[EAX]	00	000	00	01	02	03	04	05	06	07
[ECX]		001	08	09	0A	0B	0C	0D	0E	0F
[EDX]		010	10	11	12	13	14	15	16	17
[EBX]		011	18	19	1A	1B	1C	1D	1E	1F
none		100	20	21	22	23	24	25	26	27
[EBP]		101	28	29	2A	2B	2C	2D	2E	2F
[ESI]		110	30	31	32	33	34	35	36	37
[EDI]		111	38	39	3A	3B	3C	3D	3E	3F
[EAX*2]	01	000	40	41	42	43	44	45	46	47
[ECX*2]		001	48	49	4A	4B	4C	4D	4E	4F
[EDX*2]		010	50	51	52	53	54	55	56	57
[EBX*2]		011	58	59	5A	5B	5C	5D	5E	5F
none		100	60	61	62	63	64	65	66	67
[EBP*2]		101	68	69	6A	6B	6C	6D	6E	6F
[ESI*2]		110	70	71	72	73	74	75	76	77
[EDI*2]		111	78	79	7A	7B	7C	7D	7E	7F
[EAX*4]	10	000	80	81	82	83	84	85	86	87
[ECX*4]		001	88	89	8A	8B	8C	8D	8E	8F
[EDX*4]		010	90	91	92	93	94	95	96	97
[EBX*4]		011	98	89	9A	9B	9C	9D	9E	9F
none		100	A0	A1	A2	A3	A4	A5	A6	A7
[EBP*4]		101	A8	A9	AA	AB	AC	AD	AE	AF
[ESI*4]		110	B0	B1	B2	B3	B4	B5	B6	B7
[EDI*4]		111	B8	B9	BA	BB	BC	BD	BE	BF
[EAX*8]	11	000	C0	C1	C2	C3	C4	C5	C6	C7
[ECX*8]		001	C8	C9	CA	CB	CC	CD	CE	CF
[EDX*8]		010	D0	D1	D2	D3	D4	D5	D6	D7
[EBX*8]		011	D8	D9	DA	DB	DC	DD	DE	DF
none		100	E0	E1	E2	E3	E4	E5	E6	E7
[EBP*8]		101	E8	E9	EA	EB	EC	ED	EE	EF
[ESI*8]		110	F0	F1	F2	F3	F4	F5	F6	F7
[EDI*8]		111	F8	F9	FA	FB	FC	FD	FE	FF

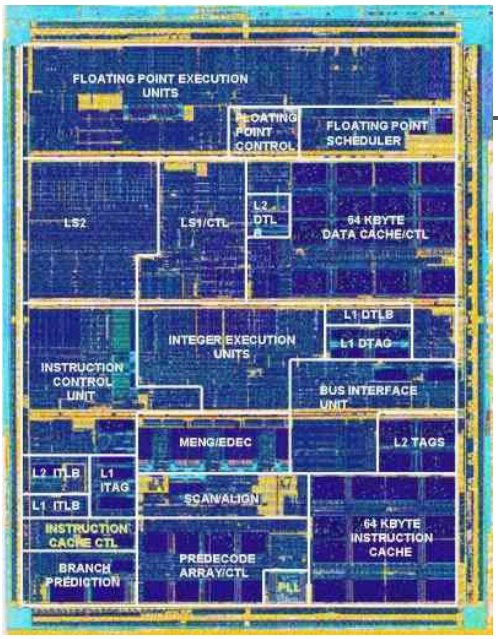
Adressierungsarten (3)

Stand Anfang 2003

- Intel
  - 32-Bit CPU
    - Pentium IV mit 3.06 GHz, Hyper-Threading
    - Xeon, Multiprozessorfähig
  - 64-Bit CPU: Itanium
- AMD
  - 32-Bit CPU
    - Athlon XP2800+ mit 2.25 GHz Taktfrequenz
    - Athlon MP 2400+ (Multiprozessor)
  - 64-Bit CPU: Hammer
- IBM
  - PowerPC970 mit 900 MHz
- … und diverse Kleine: Transmeta, VIA, SPARC, …

AMD Athlon XP

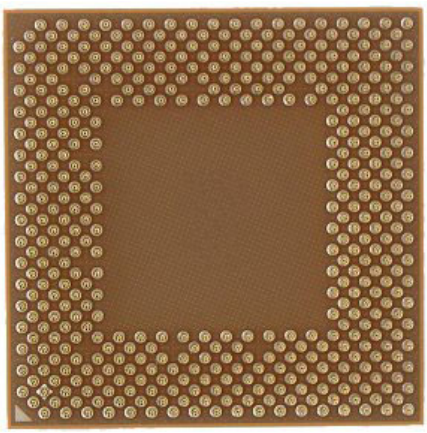
32 Bit



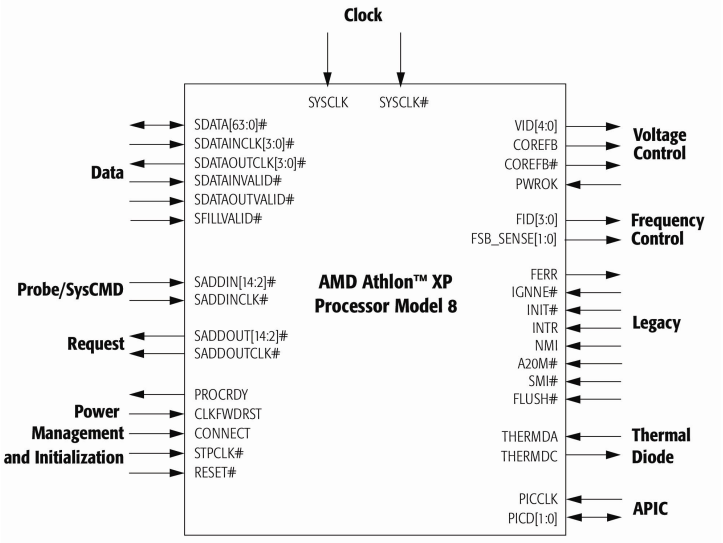
Athlon XP

- 84 mm^2
- 37.600.000 Transistoren
- 85 Grad maximale Core-Temperatur
- 68 Watt maximale Wärmeabgabe

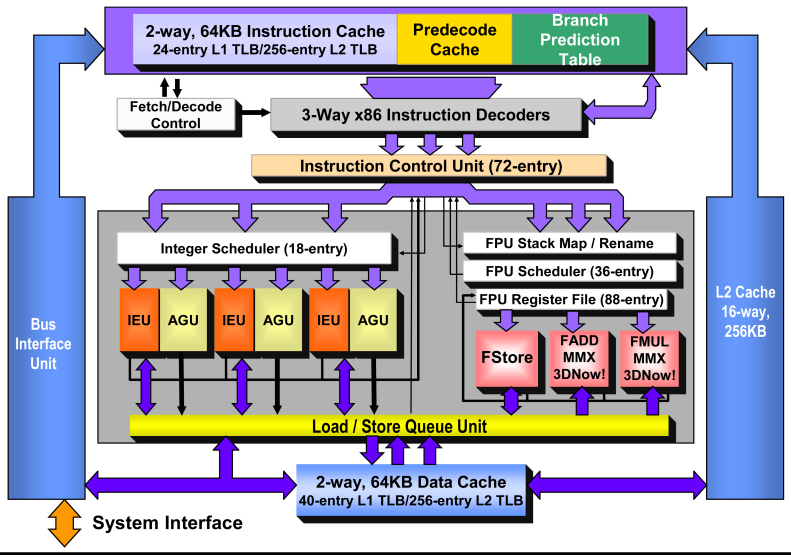
Athlon XP Chip (Gehäuse)

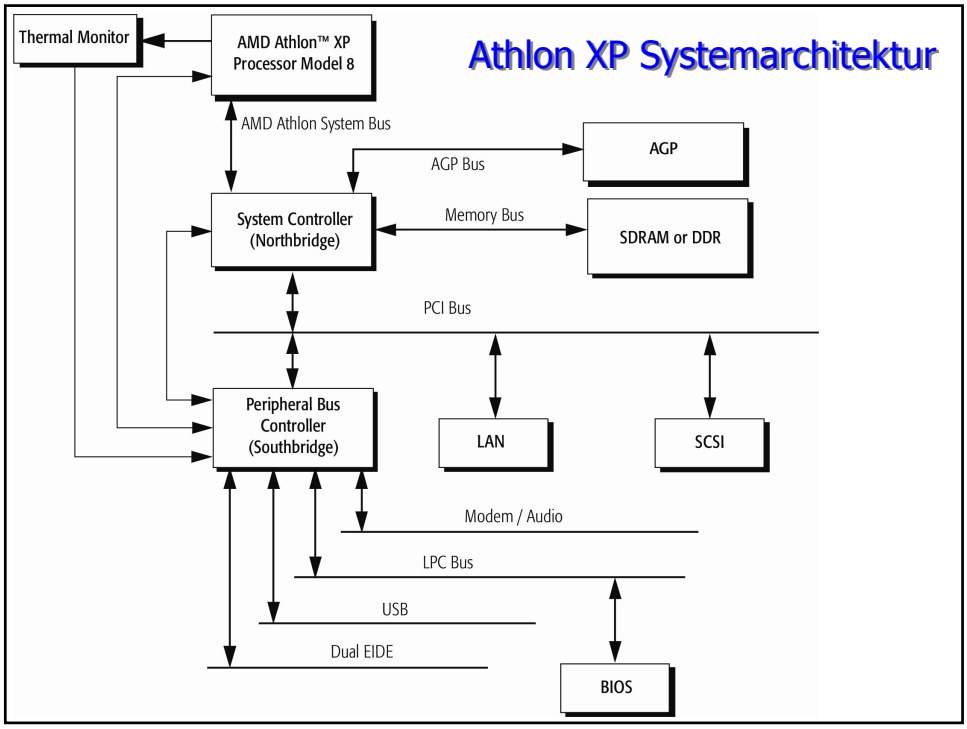


Athlon XP Chip (Logisch)



Architektur



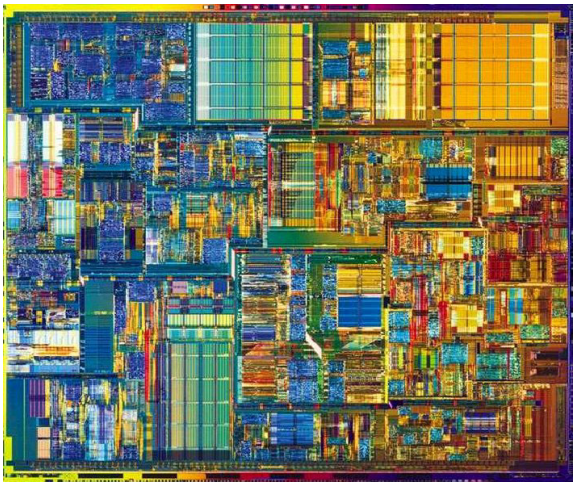


Intel Pentium IV

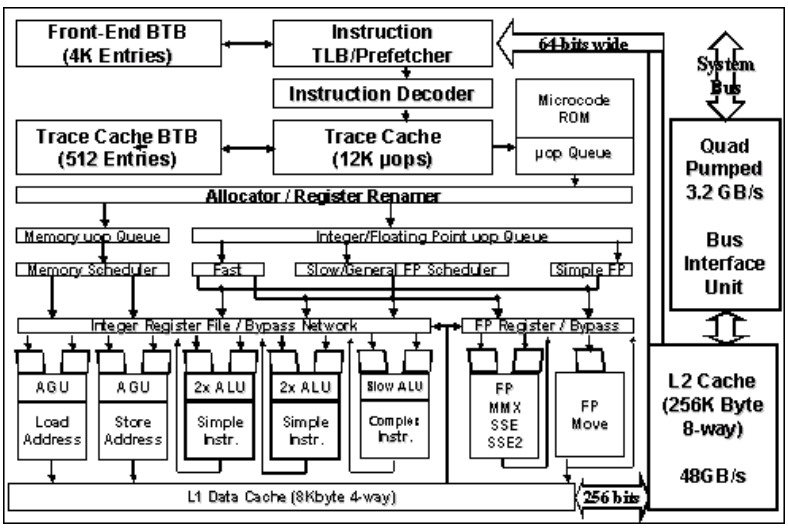
32 Bit

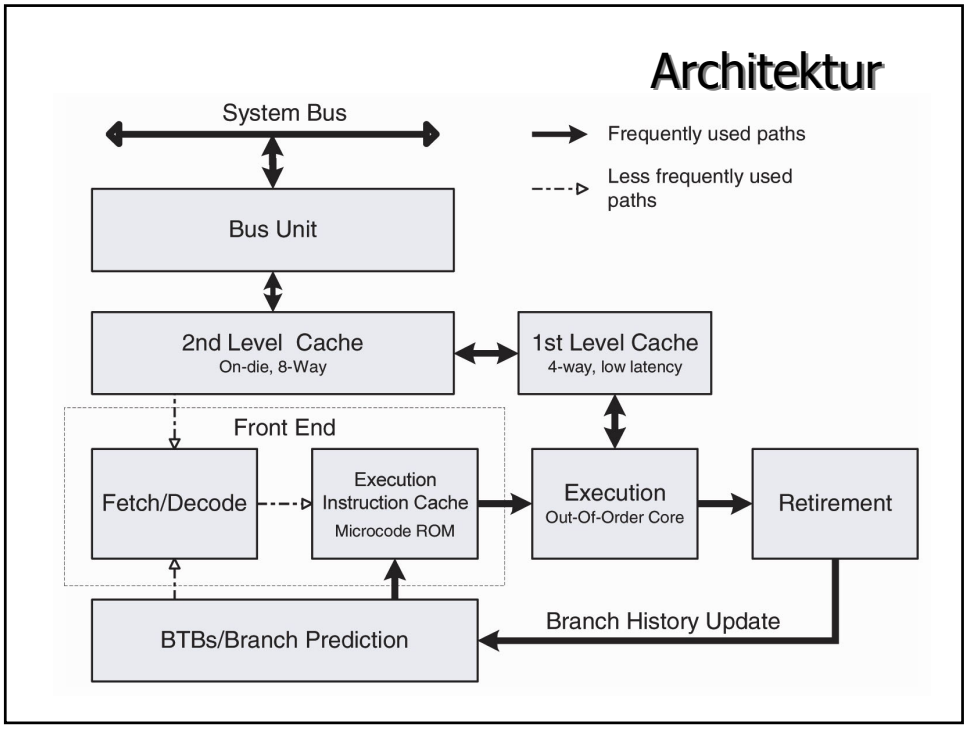
Intel Pentium IV

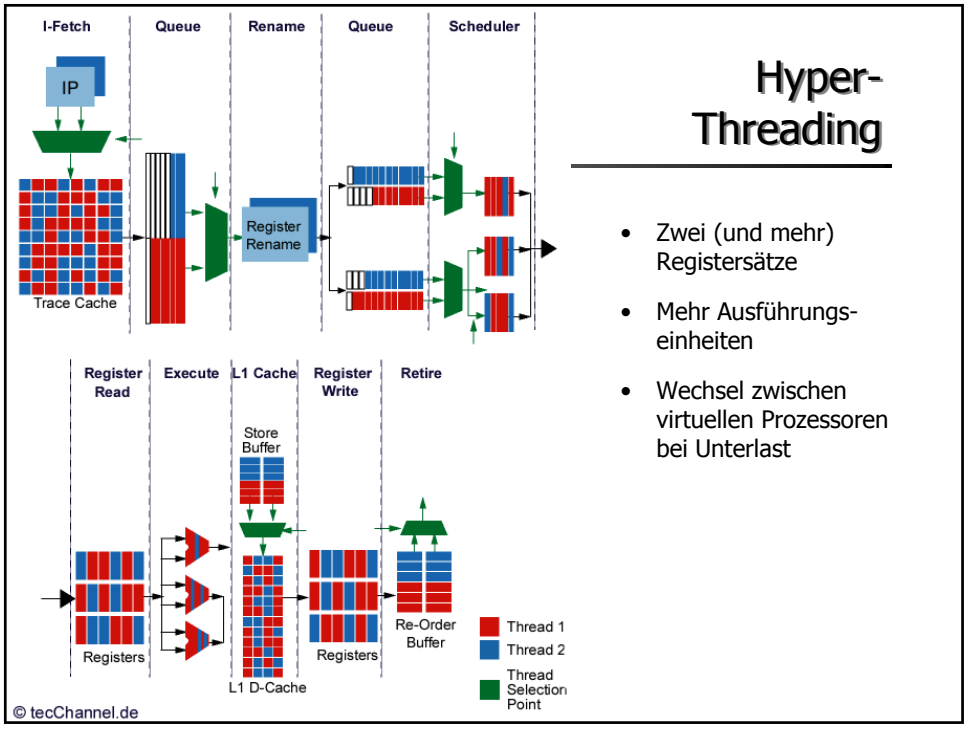
- 131 mm^2
- 55.000.000 Transistoren
- 75 Grad maximale Core-Temperatur
- 80 Watt maximale Wärmeabgabe



Pentium 4




Architektur
System Bus
Frequently used paths
Less frequently used paths
Bus Unit
2nd Level Cache
On-die, 8-Way
1st Level Cache
4-way, low latency
Front End
Fetch/Decode
Execution
Instruction Cache
Microcode ROM
Execution
Out-Of-Order Core
Retirement
BTBs/Branch Prediction
Branch History Update


I-Fetch
Queue
Rename
Queue
Scheduler
IP
Trace Cache
Register Rename
Register Read
Execute
L1 Cache
Register Write
Retire
Store Buffer
Registers
L1 D-Cache
Registers
Re-Order Buffer
Thread 1
Thread 2
Thread Selection Point
© tecChannel.de
Hyper-Threading
• Zwei (und mehr) Registersätze
• Mehr Ausführungs-einheiten
• Wechsel zwischen virtuellen Prozessoren bei Unterlast

Vergleich

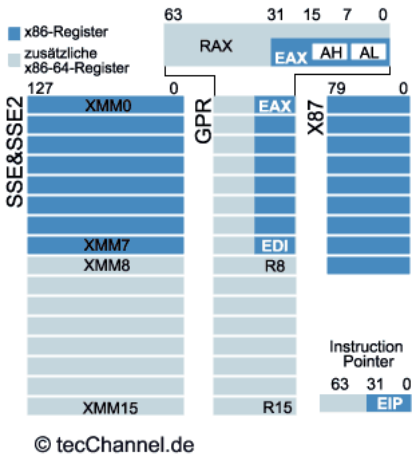
	Pentium 4 2.8GHz Northwood, stepping C1	Athlon XP 2600+ Thoroughbred, stepping 1
Core frequency	2800MHz	2133MHz
Bus frequency	533MHz (133MHz QPB)	266MHz (133MHz DDR)
Technology	0.13micron	
Cache size	L1=8+12KB, L2=512KB	L1=128KB, L2=256KB
Nominal Vcore	1.525V	1.65V
Die size	131sq.mm	84sq.mm
Number of transistors	55 million	37.6 million
Socket	Socket478	Socket A
Max. core temperature	75°C	85°C
Max. heat dissipation	80*W	68.3W
Typical heat dissipation	68.4W	62.0W

AMD Hammer

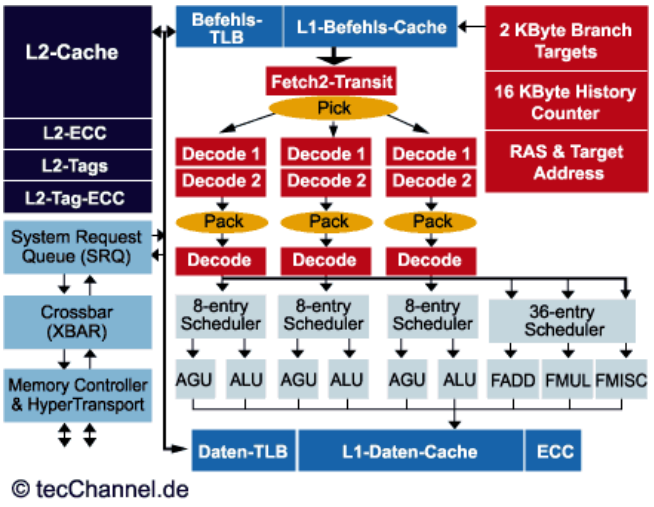
64 Bit

Beispiel AMD Hammer (64 Bit)

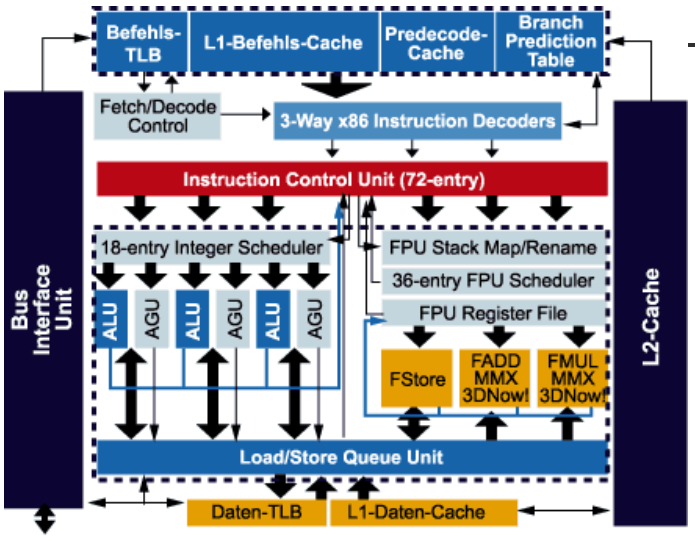
- 64-Bit-Erweiterung der IA32-Architektur
  - Abwärtskompatibel zu 32-Bit-Code
- Bekannte Einbettung der alten 32-Bit-Register
  - 128 Bit Multimedia-Register



Aufbau

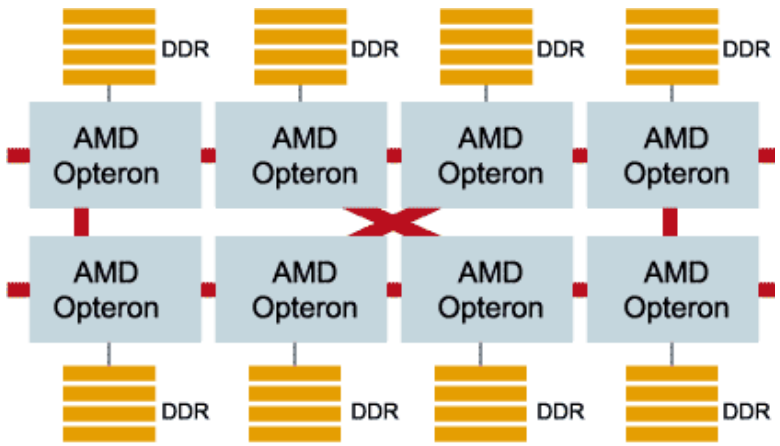


Pipeline



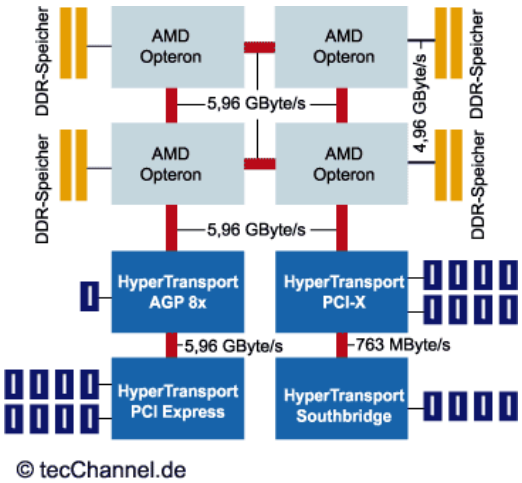
© tecChannel.de

HyperTransport zwischen CPUs

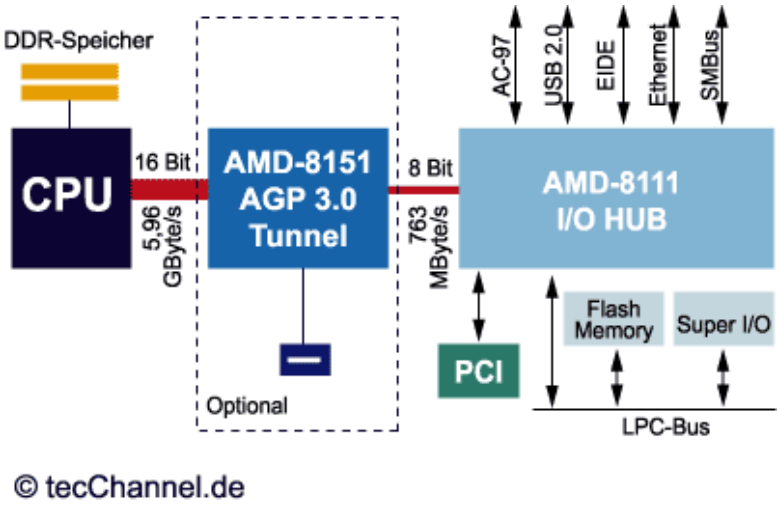


© tecChannel.de

HyperTransport zur Außenwelt



... und im Desktop-PC



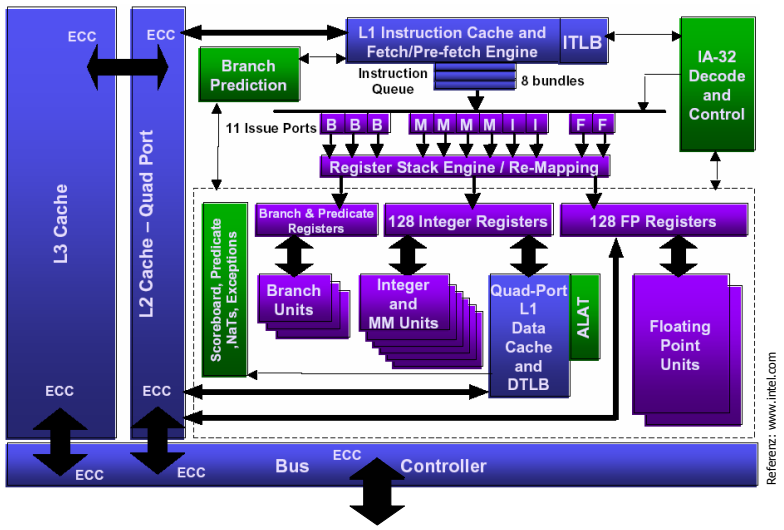
Intel Itanium 2

64 Bit

Intel IA64: Itanium 2

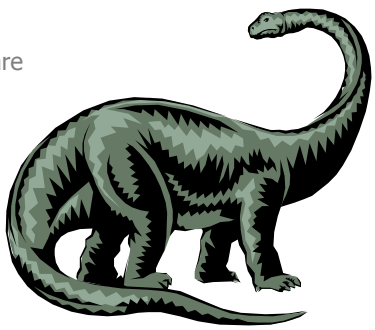
- 1 GHz Aktuelle Taktfrequenz
- Cache-Hierarchie
  - L1-Cache 32 KB
  - L2-Cache 256 KB
  - L3-Cache 1.5 MB oder 3 MB (Integriert)

Itanium 2: Architektur



CISC

- Complex Instruction Set Computer
  - Spezialregister
  - Viele Adressierungs- und Instruktionsarten
  - Instruktionen unterschiedlicher Länge
- Gründe
  - Hardware wird immer billiger, Software immer komplexer und teurer (⇒ mehr in HW lösen)
  - „Sematic Gap" zwischen HW und Compiler schließen



Die wilden 70er

- Beobachtung: Prozessoren viel zu komplex
  - Compiler nutzen nur Bruchteil der vorhandenen Instruktionen und Adressierungsarten
- Steigende Integrationsdichte
  - Was tun mit dem vielen Platz?
- Großer Leistungshunger neuer Anwendungen
  - Mikroprogramme brauchen Zeit
- Die RISC-Bewegung
  - "Make the Common Case Fast"

CISC vs. RISC

- Komplexe Instruktionen, die viele Zyklen benötigen
- (Fast) jede Instruktion kann auf Speicheroperanden zugreifen
- Kein oder wenig Pipelining
- Instruktionen werden von Mikro-programm interpretiert
- Variables Instruktionsformat
- Viele Instruktionen und Modi
- Komplexes Mikroprogramm
- Ein Registersatz

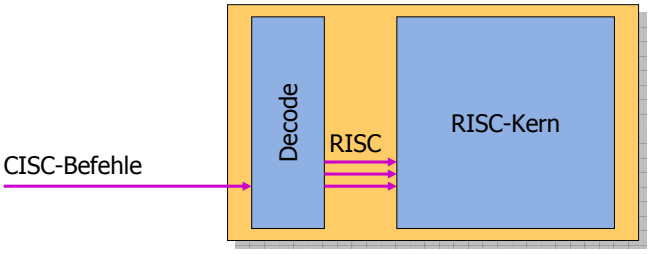
- Einfache Instruktionen, die alle nur 1 Zyklus benötigen
- Nur spezielle Instruktionen referenzieren Speicher
- Viel Pipelining
- Instruktionen werden von HW direkt ausgeführt
- Festes Instruktionsformat
- Wenig Instruktionen und Modi
- Komplexer Compiler
- Mehrere Registersätze

Bemerkungen

- 1 Instruktion pro Zyklus
  - RISC-Instruktion vergleichbar Mikroinstruktion
  - Keine RISC-Instruktion für längere Operationen (z.B. Multiplikation)
- Spezielle LOAD/STORE-Instruktionen für Speicherzugriff
  - Allgemein 1 Instruktion pro Zyklus nicht machbar
  - Problem, LOAD und STORE nicht in einem Zyklus fertig
- Pipelining
  - Abschwächung von Regel 1: 1 Instruktion pro Zyklus starten
  - Delayed Load (Store)
  - Delayed Branch
- Kein Mikroprogramm
  - Nutzung wachsender Chipflächen
  - Maximale Performanz
  - => Fester Instruktionsformat notwendig

Wölfe im Schafspelz

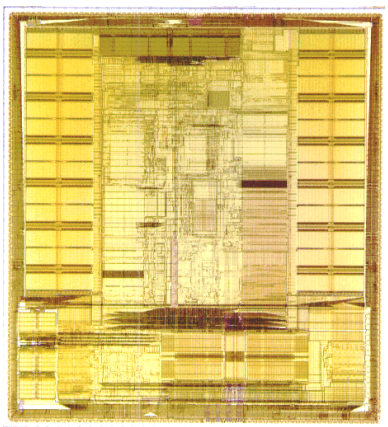
- IA32 nicht einfach ersetzbar
- CISC-Befehlssatz
- Innen meist RISC-Kern
  - Dekodieren in µOps



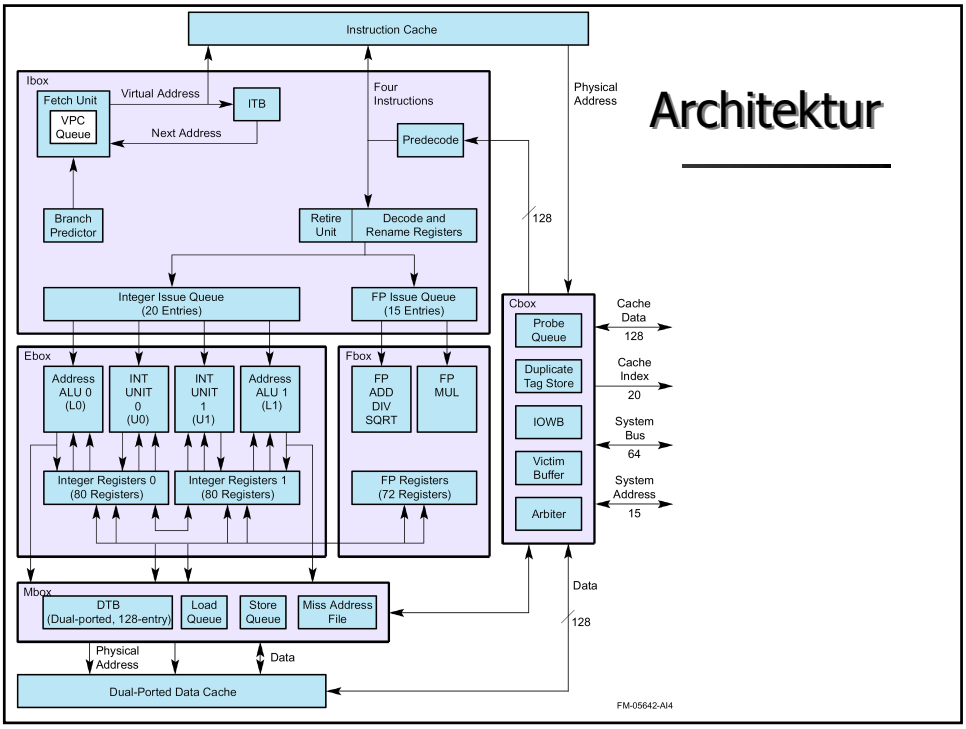
Alpha

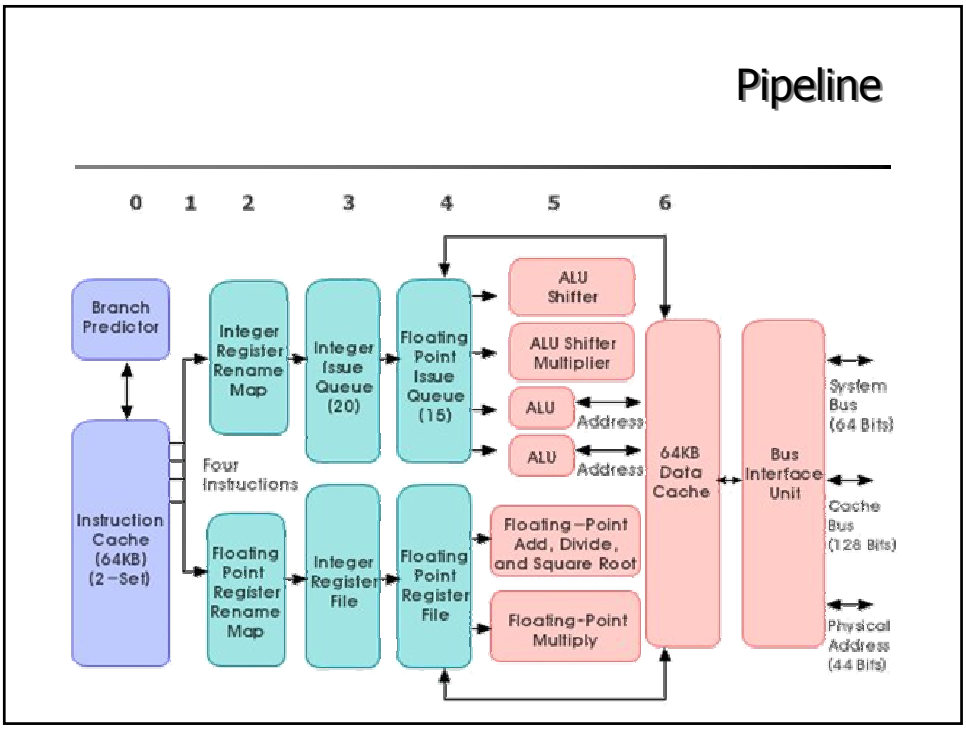
64 Bit

Alpha-Prozessor

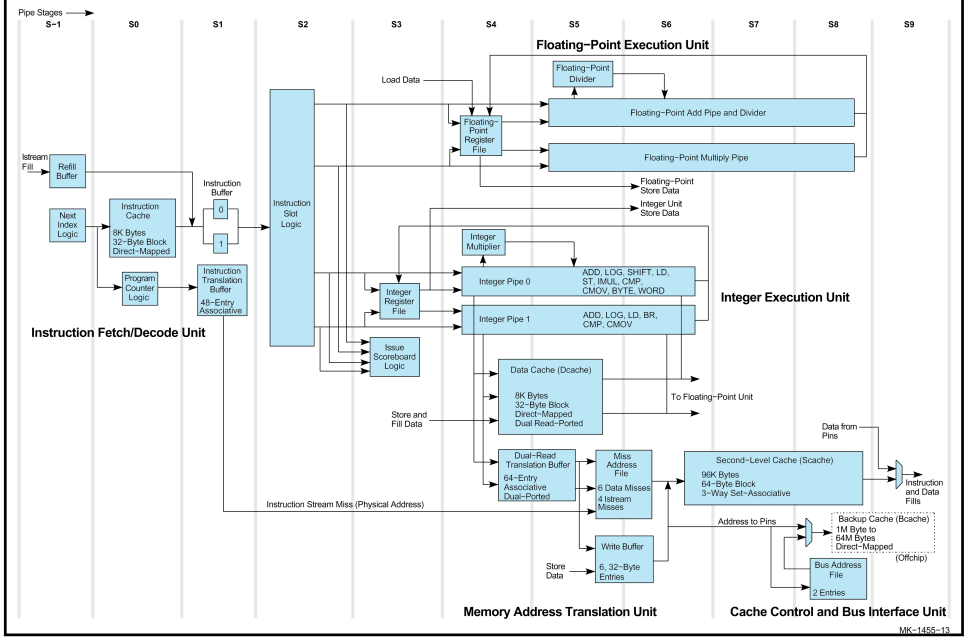


- Lange Zeit DER schnellste Prozessor der Welt
- RISC
  - 544 SpecInt2k (833 MHz)
  - 658 SpecFP2k (833 MHz)
- Specs
  - 15.2 Millionen Transistoren
  - 60 Watt
  - Fläche 3.1 cm^2
  - 35 nm
  - 588 Pin PGA

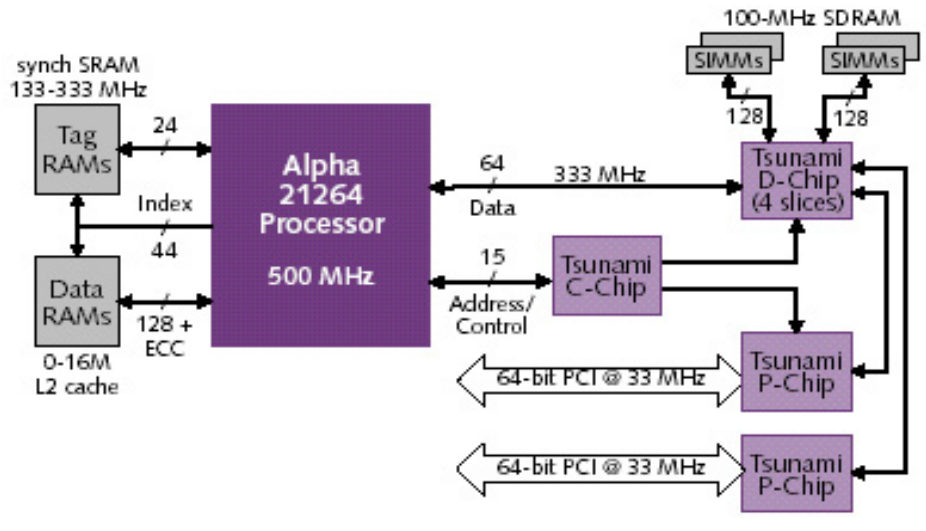

Architektur
Instruction Cache
Ibox
Fetch Unit
VPC Queue
Virtual Address
ITB
Next Address
Four Instructions
Predecode
Physical Address
Branch Predictor
Retire Unit
Decode and Rename Registers
128
Integer Issue Queue (20 Entries)
FP Issue Queue (15 Entries)
Cbox
Probe Queue
Duplicate Tag Store
IOWB
Victim Buffer
Arbiter
Cache Data
128
Cache Index
20
System Bus
64
System Address
15
Ebox
Address ALU 0 (L0)
INT UNIT 0 (U0)
INT UNIT 1 (U1)
Address ALU 1 (L1)
Integer Registers 0 (80 Registers)
Integer Registers 1 (80 Registers)
Fbox
FP ADD DIV SQRT
FP MUL
FP Registers (72 Registers)
Mbox
DTB (Dual-ported, 128-entry)
Load Queue
Store Queue
Miss Address File
Data
128
Physical Address
Data
Dual-Ported Data Cache
FM-05642-AI4


Pipeline
0 1 2 3 4 5 6
Branch Predictor
Instruction Cache (64KB) (2-Set)
Four Instructions
Integer Register Rename Map
Integer Issue Queue (20)
Floating Point Issue Queue (15)
Floating Point Register Rename Map
Integer Register File
Floating Point Register File
ALU Shifter
ALU Shifter Multiplier
ALU
Address
ALU
Address
Floating-Point Add, Divide, and Square Root
Floating-Point Multiply
64KB Data Cache
Bus Interface Unit
System Bus (64 Bits)
Cache Bus (128 Bits)
Physical Address (44 Bits)

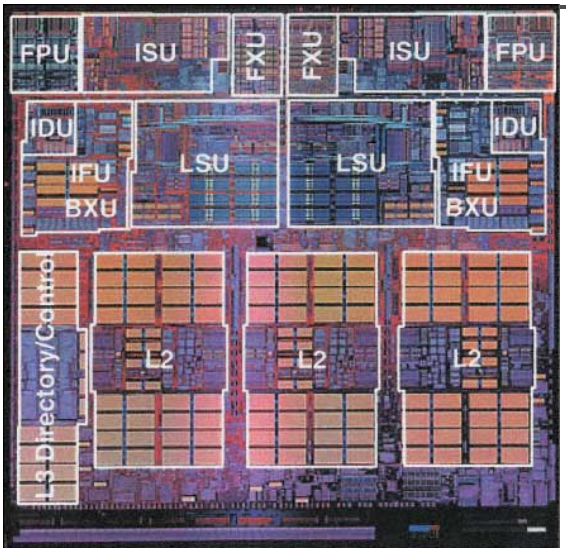
Pipeline (Detail)



Alpha-System

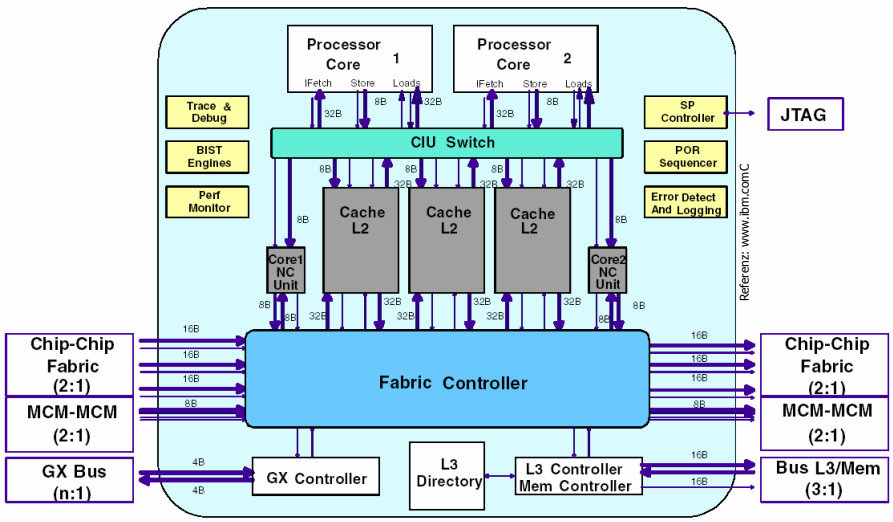


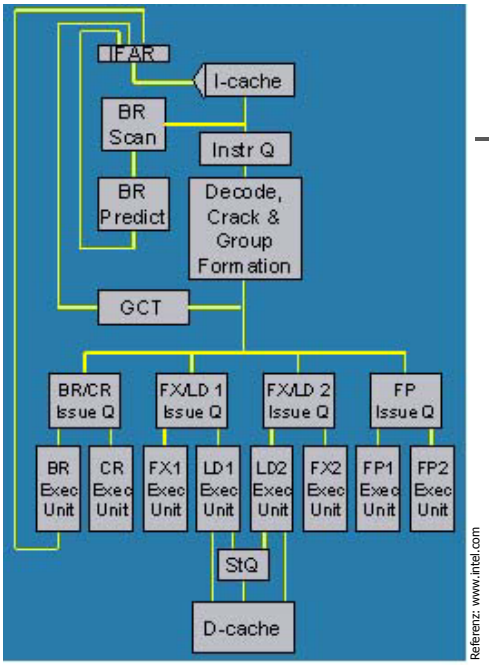
IBM Power 4



- Zwei CPU-Kerne
- 0.09 µm
- Frequenz > 1.3 GHz
- 174.000.000 Transistoren
- 115 Watt

Chip



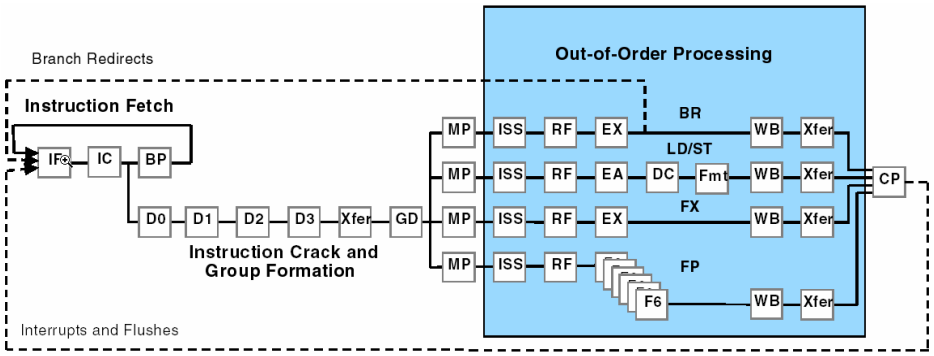


Architektur: Core

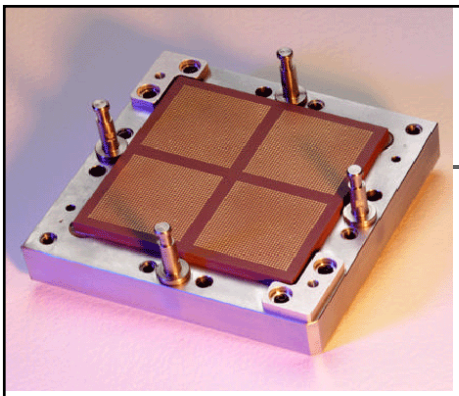
Caches

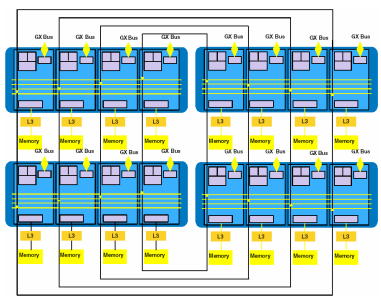
Component	Organization	Capacity per Chip
L1 Instruction Cache	Direct map, 128-byte line managed as 4 32-byte sectors	128 KB (64 KB per processor)
L1 Data Cache	2-way, 128-byte line	64 KB (32 KB per processor)
L2	8-way, 128-byte line	~ 1.5 MB
L3	8-way, 512-byte line managed as 4 128-byte sectors	32 MB
Memory	---	0-16 GB

Pipeline



Skalierbarkeit



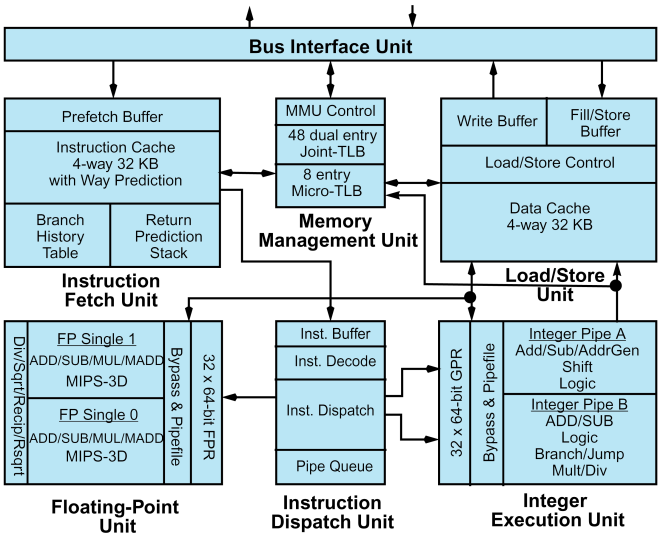


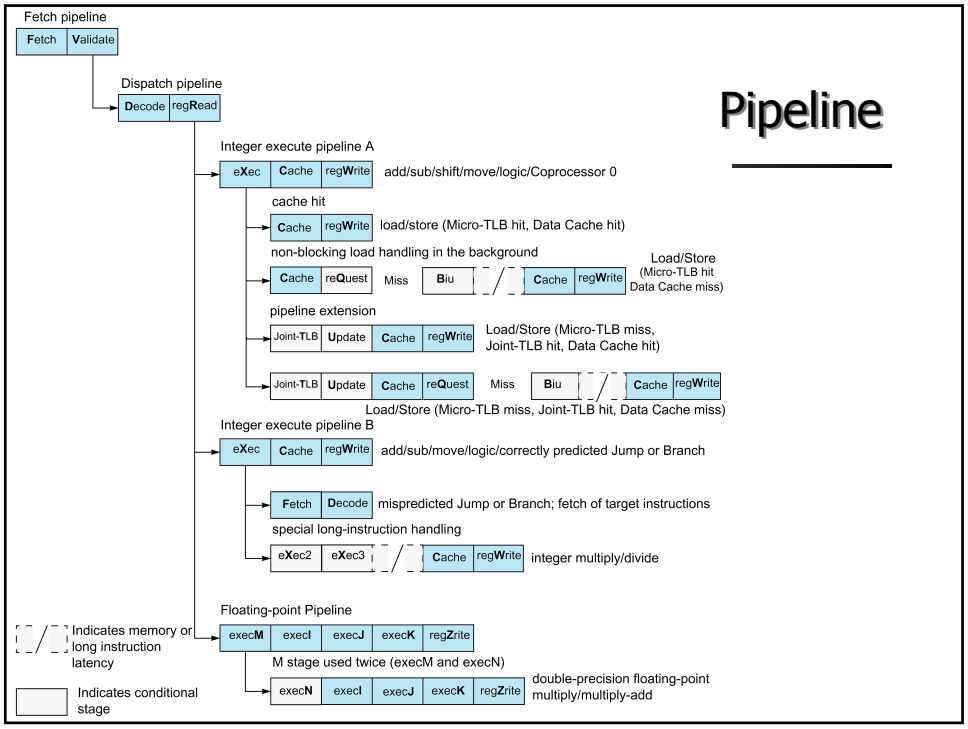
- 2 CPU pro Chip
- Multi-Chip-Module (MCM)
  - 4 Chips
  - 8 fach SMP
- MCM-to-MCM
  - 4 MCMs
  - 32 fach SMP
- Beispiel Cheetah
  (Oak Ridge National Lab, ORNL)
  - 768 CPUs
  - 1 TByte Speicher
  - 24 TB Disk
  - 4 TFLOP/s

MIPS64 20Kc

64 Bit

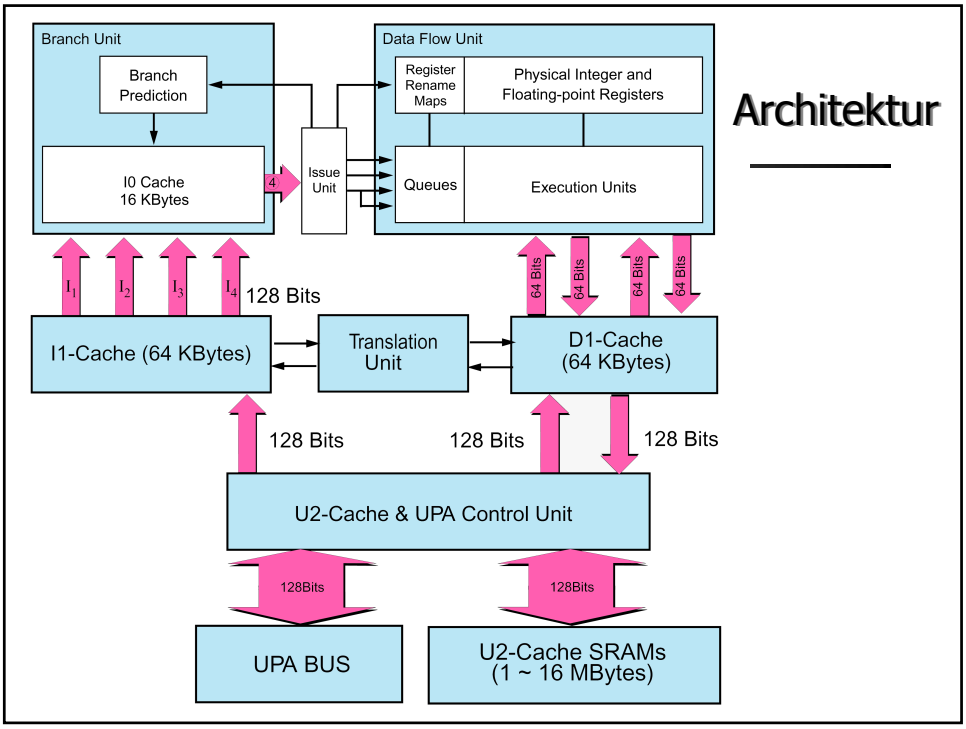
Architektur

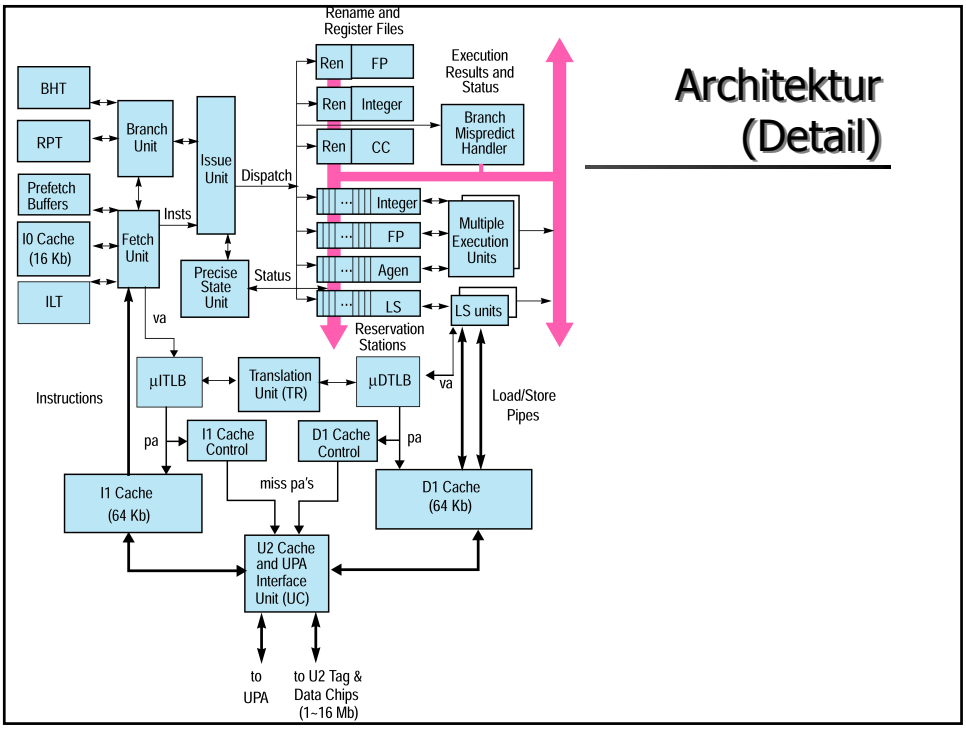



Pipeline
Fetch pipeline
Fetch
Validate
Dispatch pipeline
Decode
regRead
Integer execute pipeline A
eXec
Cache
regWrite
add/sub/shift/move/logic/Coprocessor 0
cache hit
load/store (Micro-TLB hit, Data Cache hit)
non-blocking load handling in the background
reQuest
Miss
Biu
Load/Store (Micro-TLB hit Data Cache miss)
pipeline extension
Joint-TLB
Update
Load/Store (Micro-TLB miss, Joint-TLB hit, Data Cache hit)
Load/Store (Micro-TLB miss, Joint-TLB hit, Data Cache miss)
Integer execute pipeline B
add/sub/move/logic/correctly predicted Jump or Branch
Fetch
Decode
mispredicted Jump or Branch; fetch of target instructions
special long-instruction handling
eXec2
eXec3
integer multiply/divide
Floating-point Pipeline
execM
execI
execJ
execK
regZrite
M stage used twice (execM and execN)
execN
double-precision floating-point multiply/multiply-add
Indicates memory or long instruction latency
Indicates conditional stage

SPARC

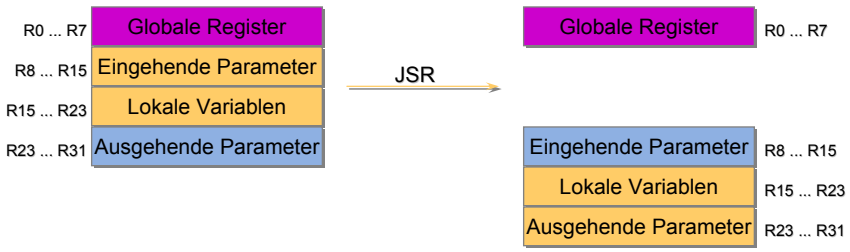
64 Bit


Architektur
Branch Unit
Branch Prediction
I0 Cache 16 KBytes
4
Issue Unit
Data Flow Unit
Register Rename Maps
Physical Integer and Floating-point Registers
Queues
Execution Units
I1
I2
I3
I4
128 Bits
64 Bits
64 Bits
64 Bits
64 Bits
I1-Cache (64 KBytes)
Translation Unit
D1-Cache (64 KBytes)
128 Bits
128 Bits
128 Bits
U2-Cache & UPA Control Unit
128Bits
128Bits
UPA BUS
U2-Cache SRAMs (1 ~ 16 MBytes)


Architektur (Detail)
Rename and Register Files
Ren
FP
Ren
Integer
Ren
CC
Execution Results and Status
Branch Mispredict Handler
BHT
RPT
Prefetch Buffers
I0 Cache (16 Kb)
ILT
Branch Unit
Issue Unit
Dispatch
Insts
Fetch Unit
Precise State Unit
Status
va
Integer
FP
Agen
LS
Multiple Execution Units
LS units
Reservation Stations
Instructions
µITLB
Translation Unit (TR)
µDTLB
va
Load/Store Pipes
pa
I1 Cache Control
D1 Cache Control
pa
miss pa's
I1 Cache (64 Kb)
D1 Cache (64 Kb)
U2 Cache and UPA Interface Unit (UC)
to UPA
to U2 Tag & Data Chips (1~16 Mb)

Überlappende Registerfenster

- Besondere Unterstützung für Unterprogramme
- Empirische Daten
  - Wenige Argumente pro Prozeduraufruf
  - Wenige lokale Variablen
  - Selten tiefere Verschachtelung
- Ein sehr großer Registersatz (128 und mehr Register)
  - Besondere Sichtbarkeitsregeln



Abschließende Vergleiche

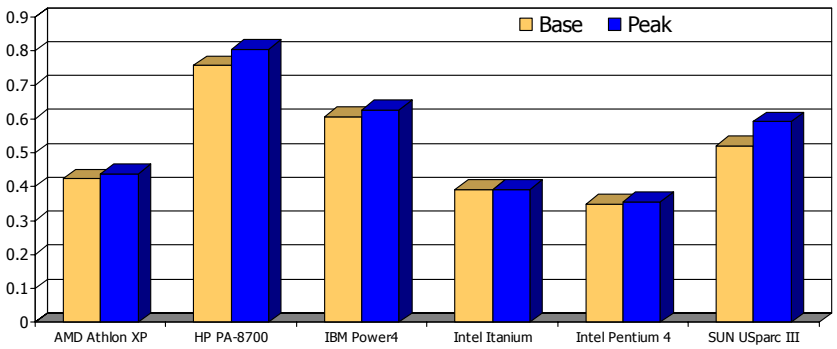
- Vergleich verschiedener 64-Bit-CPUs
- Taktfrequenz vs. Leistung
  - „Die schiere Megaherz-Zahl ist nicht alles"

Vergleich 64-Bit-Prozessoren

	McKinley	USIII	EV7	Power 4 (2 cores)
Frequency (GHz)	1.0	1.05	1.0-1.2	1.3
Pipe stages (mpb)	**8 (6)**	14 (8)	9 (~11)	12 (~11)
Sustainable Int BW	**6 / cycle**	3 / cycle	4/cycle	3/cycle
FP units	2/cycle	2/cycle	2/cycle	2/cycle
On chip cache	**3.3MB 4 arrays**	96KB 2 arrays	1.8MB 3 arrays	1.7MB 2.5 per core
D Cache read BW	**64GB/s**	16.8	19.2	41.6 / core
Die size (mm^2)	**421**	244	397	400 est.
Core size (no IO, only lowest level caches)	142	206	115	100 est.
Power (Watts)	**130**	75	125	125

Referenz: www.intel.com

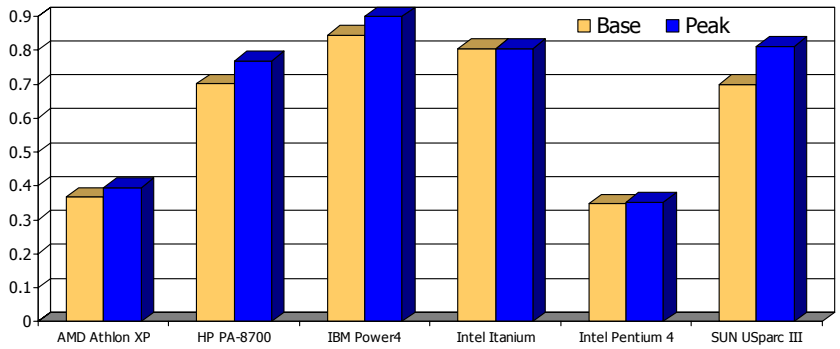
Taktfrequenz vs. Leistung (1)



- Y: Base und Peak-Performance CINT2000 / Taktfrequenz (Integer)
- Prozessoren
  - Athlon XP 1900+ (1600 MHz), HP PA-8700 (750 MHz), IBM Power4 (1CPU, 1300 MHz), Intel Itanium (800 MHz), Intel Pentium 4 (2200 GHz) und SUN UltraSPARC III-Cu (900 MHz)

Quelle: http://www.digit-life.com/articles/ibmpower4/

Taktfrequenz vs. Leistung (2)



- Y: Base und Peak-Performance CFP2000 / Taktfrequenz (Floating Point)
- Prozessoren
  - Athlon XP 1900+ (1600 MHz), HP PA-8700 (750 MHz), IBM Power4 (1CPU, 1300 MHz), Intel Itanium (800 MHz), Intel Pentium 4 (2200 GHz) und SUN UltraSPARC III-Cu (900 MHz)

Quelle: http://www.digit-life.com/articles/ibmpower4/