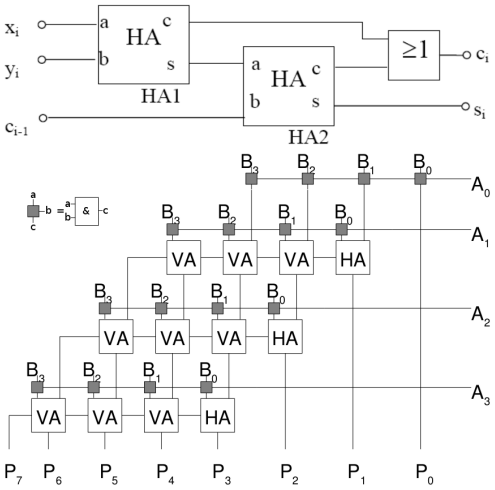
1 RA I

1.1 Darstellung von Zahlen

IEEE 754 siehe letzte Seite.

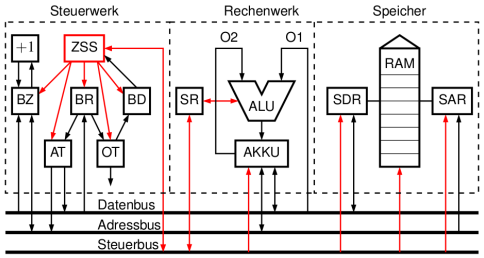
Genauigkeit

B_2/B_1	2	8	10	16
2	1	0,33	0,301	0,25
8	3	1	0,903	0,75
10	3,322	1,107	1	0,830
16	4	1,333	1,204	1



Zero-Flag Z Alle Bits $= 0$
Carry-Flag C = Auslaufender Übertag
Signed-Flag S = Höchstwertiges Bit
Overflow-Flag O Auslaufender Übertrag = einlaufender Übertrag am MSB.

1.2 von-Neumann-Architektur



Komponenten

- CPU
  - taktgesteuert
  - enthält Steuer- und Rechenwerk
  - sequentielle Befehlsabarbeitung
  - binäre Interpretation aller Signale
  - Befehlsebenenparallelität (via Pipelining, out-of-order execution, VLIW)
- Hauptspeicher
  - linear adressierbar
  - enthält Befehle und Daten
  - Zugriff über gemeinsamen Systembus
- Systembus
  - Adress-, Daten- und Steuerbus
  - 1 Teilnehmer schreibt alle anderen lesen am Bus
- Ein-/Ausgabe

Engpässe 1. Systembus als zentrales Vermittlungselement (v.N.-Flaschenhals) 2. Mehrfacher Speicherzugriff (z.B. `ID` und `WB`) 3. Keine Unterscheidung zwischen Daten und Befehlen im Speicher . Ausweg: Trennung in Programm- und Datenspeicher (Harvard-Architektur)

1.2.1 Steuerwerk

1. Zentrale Steuerschleife 2. Befehlszähler 3. Befehlsdekoder 4. Befehlsregister

Phasen

IF	ID/OF	EX/LS	WB

1.3 Typischer Befehlswortaufbau

Befehlswort			
Operationsteil	Operandenteil		
Opcode	Operand 1	Operand 2	...

1.4 Pipelining

Datenhazard Abhängigkeiten zwischen Befehlen. 1. Read-After-Write (RAW) 2. Write-After-Read (WAR) 3. Write-After-Write (WAW).. Vermeidbar mittels *Forwarding*-Register oder *Bypasses* in der ALU.

Strukturhazard Beschränkungen aufgrund von belegten Resourcen (z.B. nur ein Speicherzugriff möglich, aber `IF` und `WB` wollen auf Speicher zugreifen). Vermeidbar durch getrennten Speicher für Daten und Befehle (L1-Daten-Cache, L1-Instruction-Cache) oder Multiport-Speicher.

Kontroll-/Steuerhazard Bedingtes Sprung-/Verzweigungsziel unbekannt. Vermeidbar durch *Branch Prediction* oder Hardware *out-of-order Execution*.

1.5 Adressierungsarten

implizit Ein oder mehrere Operanden durch Befehl festgelegt (z.B. `PUSH`, `POP`, `JMP`)

überdeckt Quell-/Zieloperand gleich. (z.B. 1-Adress-Maschine mit `ADD R1`: `A` ← `A + R1`

Direkt-/Immediateoperand Operand enthält Wert statt Adresse.

direkt Operand enthält Adresse oder Registernummer.

indirekt/speicher-indirekt Operand enthält Adresse, an welcher eine weitere Adresse steht.

indiziert/register-indirekt Operand enthält Nummer eines Registers, in dem eine Adresse steht.

relativ Adresse bezieht sich auf eine bestimmte Basisadresse (z.B. `PUSH`, `POP`)

1.6 Byte-Order

Adresse	0x100	0x101	0x102	0x103
Wert	0x4F	0x73	0xC9	0x1B

Little Endian: 0x1BC9734F, Big Endian 0x4F73C91B

1.7 Cache

Arten voll-assoziativ, direct-mapped, n-Wege-Satz-assoziativ

Adressstruktur

Tag	Index	Wort	Byte

Cache-Hit-Write-Strategien

write-through 1. Cache und HS adressieren 2. Cache-Hit 3. Schreiben der Daten in den Cache 4. HS unverzüglich aktualisieren

write-back 1. Cache und HS adressieren 2. Cache-Hit 3. Schreiben der Daten in den Cache 4. Dirty-Bit setzen 5. Bei Verdrängung HS aktualisieren

Cache-Miss-Write-Strategien

write-allocate 1. Cache und HS adressieren 2. Cache-Miss 3. HS-Zugriff fortsetzen 4. Verdrängen einer Cache-Line 5. Cache-Line aus HS in Cache laden

write-around auch write-non-allocate 1. Cache und HS adressieren 2. Cache-Miss 3. HS-Zugriff fortsetzen 4. Daten in HS schreiben

Typische Verknüpfung:
write-through ↔ write-around,
write-back ↔ write-allocate.

2 RAII

2.1 RISC

- Feste Befehlslänge (⇒ einfaches Dekodieren)/Abarbeitungszeit
- fest-verdrahtete Steuerlogik
- wenige, einfache Adressierungsarten (register-indirekt, offset).
- Load-/Store-, General-Purpose-Register-Architektur
- kein direkter HS-Zugriff
- Orthogonalität der Befehle (keine überschneidende Funktionalität)
- Vorteil: Einfacher Entwurf und Herstellung
- Nachteil: Geringe Anzahl an verfügbaren Befehlen und Registern
- Nachteil: Geringere Codedichte (auch simple Befehle belegen komplettes Befehlswort)

2.2 CISC

- Variable Befehlslänge
- meist Mikroprogrammsteuerwerk
- Komplexe Adressierungsarten (speicher-indirekt, indexiert, implizit)
- Arithmetische Befehle mit Speicheroperanden

- Vorteil: Höhere Codedichte
- Nachteil: Aufwendiges Dekodieren der Befehle (Befehlslänge erkennen)
- Nachteil: unterschiedliche Befehlslaufzeiten

2.3 Klassifikation Giloi

Rechnerarchitektur

1. Hardware-Struktur
   (a) Hardwarebetriebsmittelstruktur
   - Prozessorstruktur
   - Anzahl und Aufbau der Verarbeitungseinheiten
   - FPU-Anzahl, IU-Anzahl, Pipelineaufbau

   (b) Speicherstruktur
   - pro Prozessor (Speicherhierarchie) i. Register ii. L1, L2, L3-Cache iii. Hauptspeicher iv. Festplatte v. Magnetbänder
   - Multiprozessorsysteme (MPS) i. gemeinsamer Speicher ii. verteilter Speicher

   (c) Verbindungsstruktur
   - prozessorintern (Daten-, Adress-, Steuerbus)
   - MPS externes Verbindungsnetz
   - Kooperationsregeln (Master-Slave)

2. Operationsprinzip
   (a) Informationsstruktur (Programmierschnittstelle)
   i. Klassen von Datentypen A. niedrige Elementardatentypen (Byte, Word,...) B. höherere Elementardatentypen (Floating Point) C. Gruppendatentypen (Vektoren, Felder)
   ii. Menge der Maschinendarstellung der Datentypen (z.B. *IEEE 754*)
   iii. Menge der Funktionen, die auf Datentypen angewendet werden können (≡ Befehlssatz)
   (b) Steuerungsstruktur
   i. Ablaufsteuerung A. PC-getrieben B. Datengetrieben (Datenflussrechner) C. Anforderungsgetrieben (Reduktionsarchitektur)
   ii. Datenzugriffssteuerung A. adressorientiert(von-Neumann-Variable) B. assoziativ(Caches)
   iii. Resourcenverwaltung(DMA)

2.4 Arten von Parallelität

Pipelining, Nebenläufigkeit

2.5 Eigenschaften moderner CPUs

Superskalarität Fähigkeit pro Takt (im eingelaufenen Zustand der Pipeline) mehr als eine Operation fertigstellen zu können. Realisierung durch mehrere gleiche Funktionseinheiten.

Superpipeline Weitere Unterteilung der Standard-Pipeline (z.B. bei komplizierten arithmetischen Befehlen wie `DIV`).

out-of-order execution Abarbeitung der Befehle in veränderter Reihenfolge.

in-order-completion Ergebnisse in Programmreihenfolge ausgeben (Realisierung: *Reorder Buffer*, *Renaming Register*).

2.6 Moore'sches Gesetz

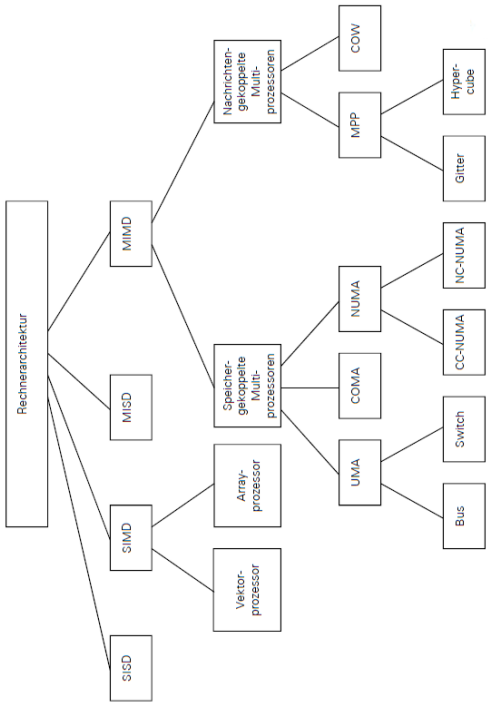
Verdopplung der Transistoren pro Chipfläche alle 18 Monate.

Auswirkungen

1. Erhöhung der Rechenleistung (a) Erhöhung der Taktfrequenz (b) Erhöhung der Anzahl der Funktionseinheiten (c) Multi-Core-Systeme
2. Behebung der Lücke zwischen Prozessor und Speicherperformance durch Caches
3. Integration von weiteren Einheiten (wie z.B. Grafikeinheiten)

2.7 Klassifizierung nach Flynn

SISD klassischer v.-N.-Rechner
SIMD Vektorrechner, Feldrechner
MIMD Multiprozessorsysteme
MISD leere Klasse oder Spezialfälle wie redundante Berechnungen
UMA Uniform Memory Access. Einheitlicher Zugriff mit derselben Bandbreite und Latenz
NUMA Non-UMA
CC-NUMA cache coherent NUMA
NC-NUMA non-cache coherent NUMA
COMA Cache-only memory architecture



2.8 Vektorrechner

- *Pipelining*
- andere Speicherorganisation (kein Cache), Daten werden aus DRAM-Speicher in Vektorregister geladen.
- arithmetisches Pipelining (z.B. bei Floating Point)
- *Chaining*: Pipeline-Verkettung
- Erweiterungen: mehre Vektorunits pro Vektorprozessor, mehrere Vektorprozessoren pro Vektorrechner
- Scalar-Unit für nicht vektorisierbare Aufgaben vorhanden, aber meist langsam.
- Vorteil: Hohe Leistung auch bei hohen Problemgrößen.
- Nachteil: Probleme müssen vektorisierbar sein.

2.9 Feldrechner

- *Nebenläufigkeit*
- sehr viele Rechenelemente (processing elements)
- ein Universalrechner steuert Verarbeitung, alle PEs machen zu einem Zeitpunkt das Gleiche (synchrone Verabeitung).

2.10 Befehlssatzerweiterungen

Prinzip: Feldrechner

MMX **M**ulti**m**edia **Ex**tensions. 64-Bit Integer-Register mit einem Befehl als 8×8-, 4×16- oder 2×32-Bit-Register angesprochen werden.

(I)SSE (**I**nternet) **S**treaming **SIMD** **Ex**tensions. 128-Bit-Floating-Point-Register kann parallel auf 4×32- oder 2×64-Bit arbeiten.

AVX **A**dvanced **V**ector **Ex**tensions. 256-Bit-Floating-Point-Register, später auch Integer.

2.11 Kennwerte

IPS Instructions per second.
IOPS Integer operations per second.
FLOPS Floating point operations per second.
IOOPS I/O operations per second.

$XPS = f \cdot XPC$
Geschwindigkeitsgewinn $S_p = \frac{T_1}{T_p}$
Parallele Effizienz $E_p = \frac{S_p}{p}$
mittlere Operationszeit $T = \sum_{i=1}^{n} t_i p_i$
$t_i \ldots$ Operationszeit des Befehls i
$p_i \ldots$ relative Häufigkeit des Befehls i
MIPS-Rate $P_{\text{MIPS}} = \frac{f}{N_i N_m}$, f in MHz.

N_i... Mittelwert der Taktzyklen pro Befehl
N_m... Speicherzugriffsfaktor
Operationsredundanz $R_p = \frac{Z_p}{Z_1}$
Z... Anzahl der Operationen
T... Zeit(-schritte)
p... Anzahl der Prozessoren
Auslastung $U_p = \frac{Z_p}{pT_p}$
Effektivität $F = \frac{S_p E_p}{T_1}$

2.12 Verbindungsnetzwerke

Statische VBN

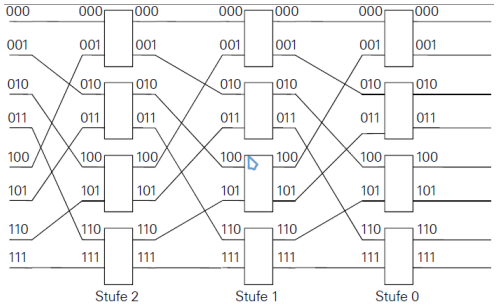
- feste Verbindungen zwischen Knoten
- Vermittlungsfunktionen sind Bestandteil des VBN
- alle Knoten bilden zusammenhängenden Graphen (z.B. Ring, Torus, 2D-Gitter, Hypercube, Stern)

Dynamische VBN Verbindungen werden zur Laufzeit geschalten. Arten 1. Bus 2. zellenbasierte Netze (*OMEGA*-Netz) 3. Kreuzschienenverteiler .

Kennwerte

Mittlerer Knotenabstand $\bar{d} = \sum_{i=1}^{k} ip_i$, p_i... Prozentualer Anteil an Knoten mit Pfadlänge i.
Durchmesser k, minimaler Abstand der am weitesten entfernten Knoten.
Grad d, Verbindungen pro Knoten
Knotenanzahl N
Halbierungsbreite kg, minimale Anzahl der Verbindungen, die aufgetrennt werden müssen, um das Netz in zwei *gleich große* Teile zu teilen.
Bisektionsbandbreite w = Halbierungsbreite · Übertragungsbandbreite pro Verbindung.
kleinste Erweiterung Anzahl der Knoten, die hinzugefügt werden müssen für strukturerhaltende Erweiterung.
Konnektivität Minimum aus Knoten- und Kantenkonnektivität. Anzahl der Knoten bzw. Kanten, die entfernt werden müssen, um das Netz beliebig zu teilen.
Einbettung Eignung der Topologie in andere Topologien eingebettet werden zu können.

2.12.1 OMEGA-Netzwerk



Prinzip: *Perfect Shuffle*, benötigt $\operatorname{ld} N$ Schaltstufen.

Routing über Zieladresse

0 unterer Ausgang, 1 oberer Ausgang

Routing über `XOR`-Verknüpfung `XOR`-Verknüpfung von Ziel- und Quelladresse. 0 durchschalten, 1 umschalten.

Blockierung Blockierung bei $\operatorname{ld} N$ Übereinstimmungen.

2.13 Leitungsvermittlung

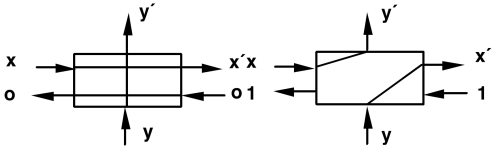
- Aufbau von fester Leitung für die Dauer der Verbindung
- Vorteil: wenn Leitung steht, schnelle Verbindung
- Nachteile: 1. Alle Knoten in der Leitung sind für andere blockiert. 2. Große Aufbauzeit (Latency)
- Einsatz: wenig Kommunikation, aber datenintensiv

2.14 Paketvermittlung

Nachricht wird in Pakete unterteilt und von Teilnehmer zu Teilnehmer geschickt. Es sind Zusatzinformationen (z.B. Quell-/Zieladresse, Paketnr., Nachspann (Prüfsumme)).
Arten

store and forward Nachricht wird von Knoten zu Knoten gesendet ⇐ großer Speicherbedarf pro Knoten.
wormhole routing Sender hört auf zu Senden, wenn Nachbar blockiert. Nachricht liegt wie ein Wurm im Netzwerk.
virtual cut-through Sender sendet bei Blockierung weiter. Extremfall: Knoten vor Blockierung muss gesamte Nachricht speichern.

2.15 Systolische Arrays



SIMD, wie Feldrechner bzgl. viele Verarbeitungseinheiten; wie Vektorrechner bzgl. verallgemeinertem Pipelining-Prinzip.

2.16 MESI

Cache-Kohärenz-Protokoll

2.16.1 Anforderungen

Datenkohärenz Gewährleistung, dass kein Prozessor auf veraltete Daten zugreifen kann.

- ohne Cache: kein Problem, da nur ein Speicher.
- mit Cache: schwierig, Kohärenz-Protokoll erforderlich

Skalierbarkeit Verhalten des Systems beim Hinzufügen zusätzlicher Prozessoren.

- ohne Cache: schwierig, Verengung v.N.-Flaschenhals, Grenze 32 Prozessoren.
- mit Cache: besser, viele Berechnungen können mit eigenen Caches durchgeführt werden ⇐ Entlastung Hauptspeicher.

2.16.2 Zustände

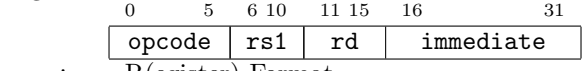
Exclusive *M*odified Die Zeile befindet sich exklusiv in diesem Cache und wurde modifiziert.
***E*xclusive Unmodified** Die Zeile befindet sich exklusiv in diesem Cache und wurde nicht modifiziert.
***S*hared Unmodified** Die Zeile befindet sich noch in einem anderen Cache und wurde nicht modifiziert.
***I*nvalid** Die Zeile ist ungültig.

2.16.3 *Snooping-Bus* mit Steuersignalen

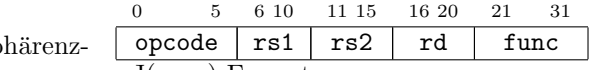
Invalid-Signal Invalidierung von Cache-Zeilen in anderen Caches.
Shared-Signal Andere Prozessoren zeigen damit an, dass die zu ladende Cachezeile bei ihnen bereits als Kopie vorhanden ist.
Retry-Signal Damit kann ein anderer Prozessor aufgefordert werden, das Laden eines Blocks abzubrechen, um es später wieder aufzunehmen nachdem der auffordernde Prozessor diesen Block aus seinem Cache in den HS zurückgeschrieben hat.

2.17 Befehlsformate der DLX-Architektur

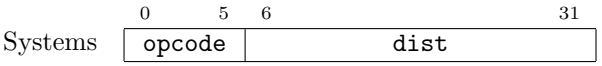
I(mmediate)-Format



R(egister)-Format



J(ump)-Format



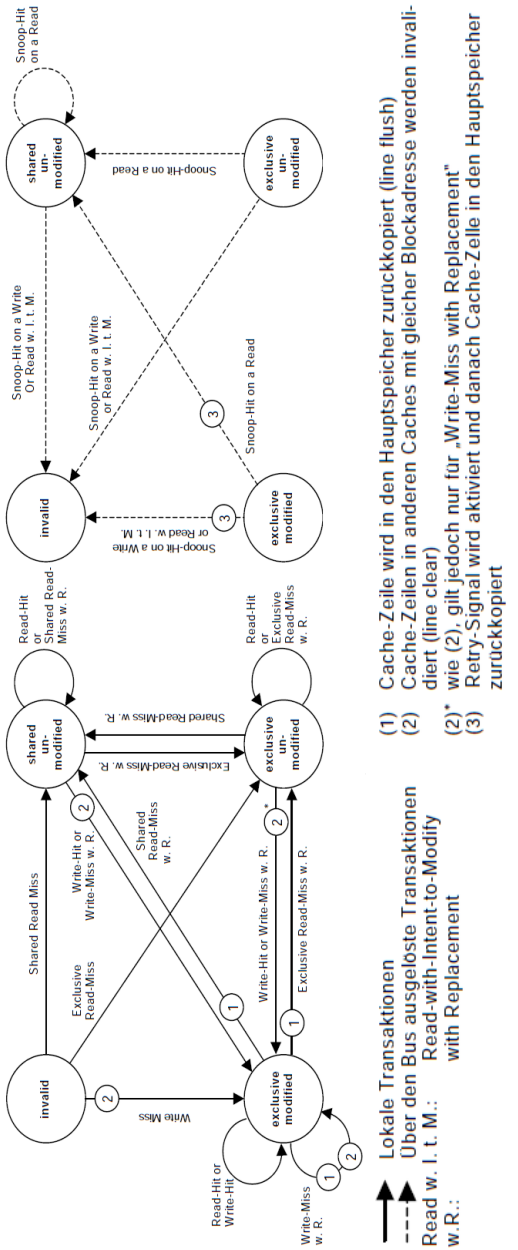
3 Tabellen

3.1 Boolesche Grundfunktionen

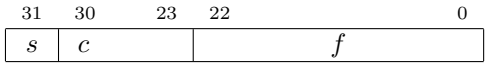
x_1, x_2	0, 0	0, 1	1, 0	1, 1	Funktion		Name
f_0	0	0	0	0	$x_1 \cdot \bar{x_1}$	0	Kontradiktion, Nullfunktion
f_1	0	0	0	1	$x_1 \cdot x_2$	$\lfloor x_1, x_2 \rfloor$	Konjunktion, AND(x_1, x_2)
f_2	0	0	1	0	$x_1 \cdot \bar{x_2}$	$x_1 > x_2$	Inhibition von x_1
f_3	0	0	1	1	x_1	x_1	Identität von x_1
f_4	0	1	0	0	$\bar{x_1} \cdot x_2$	$x_1 < x_2$	Inhibition von x_2
f_5	0	1	0	1	x_2	x_2	Identität von x_2
f_6	0	1	1	0	$(x_1 \cdot \bar{x_2}) + (\bar{x_1} \cdot x_2)$	$x_1 \neq x_2$	Antivalenz, Alternative, XOR(x_1, x_2)
f_7	0	1	1	1	$x_1 + x_2$	$\lceil x_1, x_2 \rceil$	Disjunktion, OR(x_1, x_2)
f_8	1	0	0	0	$x_1 \bar{+} x_2 = \bar{x_1} \cdot \bar{x_2}$	$1 - \lceil x_1, x_2 \rceil$	Peirce-Funktion, NOR(x_1, x_2)
f_9	1	0	0	1	$(x_1 \cdot x_2) + (\bar{x_1} \cdot \bar{x_2})$	$x_1 = x_2$	Äquivalenz
f_{10}	1	0	1	0	$\bar{x_2}$	$1 - x_2$	Negation von x_2, NOT(x_2)
f_{11}	1	0	1	1	$x_1 + \bar{x_2}$	$x_1 \geq x_2$	Replikation
f_{12}	1	1	0	0	$\bar{x_1}$	$1 - x_1$	Negation von x_1, NOT(x_1)
f_{13}	1	1	0	1	$\bar{x_1} + x_2$	$x_1 \leq x_2$	Implikation
f_{14}	1	1	1	0	$x_1 \bar{\cdot} x_2 = \bar{x_1} + \bar{x_2}$	$1 - \lfloor x_1, x_2 \rfloor$	Sheffer-Funktion, NAND(x_1, x_2)
f_{15}	1	1	1	1	$x_1 + \bar{x_1}$	1	Tautologie, Einsfunktion

3.2 Hardware Description Notation (HDN)

HDN	Beispiel	Beschreibung
$\leftarrow$	$R1 \leftarrow R2$	Transfer, logisch
M	$R1 \leftarrow M[x]$	Speicherzugriff, byteweise
$GPR[x]$	$GPR[rd]$	Zugriff auf GPR
$\leftarrow_n$	$R1 \leftarrow_{16} M[x]\#\#M[x+1]$	Transfer mit Länge
X_n	$R1_0$	Zugriff auf Einzelbit
$X_{n...m}$	$R1_{16...31}$	Zugriff auf Bitkette
X^n	$R1_{16}^{16}$	Wiederholung
$\#\#$	$M[X]\#\#M[x+1]$	Verkettung
$\ll$	$R1 \ll 2$	Bitverschiebung
$\gg$	$R1 \gg 2$	Bitverschiebung
$\gg_a$	$R1 \gg_a 2$	Bitverschiebung, arithmetisch
== != < > >= <=		Relationale Operatoren
& | ^ !		Logische Operatoren
+ − * /		Arithmetische Operatoren



3.3 IEEE 754 (single precision)



s Vorzeichenbit, c Charakteristik mit $c = e + B$ für Exponent e und Bias-Konstante $B = 127 = 0111\ 1111_2$, f *Fractional Part* für Mantisse $m = 1, f$ oder $m = 0, f$

$0 < c < 2B + 1$	f beliebig	$Z = (-1)^s \cdot (1, f) \cdot 2^{c-B}$	normalisiert
$c = 0$	f beliebig	$Z = (-1)^s \cdot (0, f) \cdot 2^{1-B}$	denormalisiert
$c = 2B + 1$	$f = 0$	$Z = (-1)^s \infty$	unendlich
$c = 2B + 1$	$f \neq 0$	$Z =$ `NaN`	Not a Number