Spezifikation der ISA

Jeder Befehl besteht aus einem 6 bits langen Opcode am Ende des Befehlsworts, d.h. auf den niedrigstwertigen 6 bits. Dann folgen die Operanden in der jeweils angegebenen Reihenfolge in Richtung steigender Wertigkeit der bits. Registeroperanden haben stets jeweils 5 bits. Immediate-Operanden stehen stets am höchstwertigen Ende des Befehlswortes und nehmen dort die maximale Restanzahl bits in Anspruch. Operanden werden folgendermaßen benannt: `R` für Register, `Im` für Immediate. Die Zahl dahinter gibt die Position des Parameters an.

Befehlsname	**Opcode**	**Operanden**	**Semantik**
`ADD`	`10000 0`	`R1, R2, R3`	`R1 := R2 + R3`
`ADDI`	`10000 1`	`R1, R2, Im`	`R1 := R2 + Im`
`SUB`	`10001 0`	`R1, R2, R3`	`R1 := R2 - R3`
`SUBI`	`10001 1`	`R1, R2, Im`	`R1 := R2 - Im`
`AND`	`10010 0`	`R1, R2, R3`	`R1 := R2 & R3`, wobei `&` bitweises Und bezeichnet
`ANDI`	`10010 1`	`R1, R2, Im`	`R1 := R2 & Im`, wobei `&` bitweises Und bezeichnet
`OR`	`10011 0`	`R1, R2, R3`	`R1 := R2 | R3`, wobei `|` bitweises Oder bezeichnet
`ORI`	`10011 1`	`R1, R2, Im`	`R1 := R2 | Im`, wobei `|` bitweises Oder bezeichnet
`NOT`	`10100 0`	`R1, R2`	`R1 := ! R2`, wobei `!` das bitweise Nicht bezeichnet
`SHL`	`10101 0`	`R1, R2, R3`	`R1 := R2 << R3`, wobei << für bitweise Linksverschiebung steht
`SHLI`	`10101 1`	`R1, R2, Im`	`R1 := R2 << Im`, wobei << für bitweise Linksverschiebung steht
`SHRA`	`10110 0`	`R1, R2, R3`	`R1 := R2 >> R3`, wobei >> für bitweise arithmetische Rechtsverschiebung steht (die höchstwertigen Bits von `R1` werden mit dem Wert des höchstwertigen Bits von `R2` aufgefüllt)
`SHRL`	`10111 0`	`R1, R2, R3`	`R1 := R2 >>> R3`, wobei >>> für bitweise logische Rechtsverschiebung steht (die höchstwertigen Bits von `R1` werden mit `0` aufgefüllt)
`SHRAI`	`10110 1`	`R1, R2, Im`	`R1 := R2 >> Im`, wobei >> für bitweise arithmetische Rechtsverschiebung steht (die höchstwertigen Bits von `R1` werden mit dem Wert des höchstwertigen Bits von `R2` aufgefüllt)
`SHRLI`	`10111 1`	`R1, R2, Im`	`R1 := R2 >>> Im`, wobei >>> für bitweise logische Rechtsverschiebung steht (die höchstwertigen Bits von `R1` werden mit `0` aufgefüllt)
`JMPA`	`00000 0`	`R1`	`PC := R1`, wobei `PC` den Program

			Counter bezeichnet
JMPR	00000 1	Im	PC := PC + Im, wobei PC den Program Counter bezeichnet
BRA	00001 0	R1	PC := R1, falls das TRUE-Flag gesetzt ist; andernfalls keine Auswirkung
BRR	00001 1	Im	PC := PC + Im, falls das TRUE-Flag gesetzt ist; andernfalls keine Auswirkung
CEQ	00010 0	R1, R2	Falls in R1 und R2 identische Bitmuster stehen, wird das TRUE-Flag gesetzt, andernfalls wird es zurückgesetzt
CEQI	00010 1	R1, Im	Falls in R1 das Bitmuster Im steht, wird das TRUE-Flag gesetzt, andernfalls wird es zurückgesetzt
CLTU	00011 0	R1, R2	Falls in R1 ein Bitmuster steht, das als vorzeichenlose Ganzzahl interpretiert kleiner ist als das entsprechend interpretierte Bitmuster in R2, wird das TRUE-Flag gesetzt, sonst wird es zurückgesetzt.
CLTS	00100 0	R1, R2	Falls in R1 ein Bitmuster steht, das als vorzeichenbehaftete Ganzzahl interpretiert kleiner ist als das entsprechend interpretierte Bitmuster in R2, wird das TRUE-Flag gesetzt, sonst wird es zurückgesetzt.
CLTUI	00011 1	R1, Im	Falls in R1 ein Bitmuster steht, das als vorzeichenlose Ganzzahl interpretiert kleiner ist als das entsprechend interpretierte Bitmuster Im, wird das TRUE-Flag gesetzt, sonst wird es zurückgesetzt.
CLTSI	00100 1	R1, Im	Falls in R1 ein Bitmuster steht, das als vorzeichenbehaftete Ganzzahl interpretiert kleiner ist als das entsprechend interpretierte Bitmuster Im, wird das TRUE-Flag gesetzt, sonst wird es zurückgesetzt.
CGTU	00101 0	R1, R2	Falls in R1 ein Bitmuster steht, das als vorzeichenlose Ganzzahl interpretiert größer ist als das entsprechend interpretierte Bitmuster in R2, wird das TRUE-Flag gesetzt, sonst wird es zurückgesetzt.
CGTS	00110 0	R1, R2	Falls in R1 ein Bitmuster steht, das als vorzeichenbehaftete Ganzzahl interpretiert größer ist als das entsprechend interpretierte Bitmuster in R2, wird das TRUE-Flag gesetzt, sonst wird es zurückgesetzt.
CGTUI	00101 1	R1, Im	Falls in R1 ein Bitmuster steht, das als vorzeichenlose Ganzzahl interpretiert größer ist als das entsprechend

			interpretierte Bitmuster `Im`, wird das `TRUE`-Flag gesetzt, sonst wird es zurückgesetzt.
`CGTSI`	`00110 1`	`R1, Im`	Falls in `R1` ein Bitmuster steht, das als vorzeichenbehaftete Ganzzahl interpretiert größer ist als das entsprechend interpretierte Bitmuster `Im`, wird das `TRUE`-Flag gesetzt, sonst wird es zurückgesetzt.
`MOVE`	`00111 0`	`R1, R2`	`R1 := R2`
`MOVI`	`00111 1`	`R1, Im`	`R1 := Im`
`LOAD`	`010000`	`R1, R2, Im`	`R1 := SPEICHER[R2 + Im]`
`STORE`	`010001`	`R1, R2, Im`	`SPEICHER[R2 + Im] := R1`
`NOP`	`010010`		keine Auswirkung
`HALT`	`010011`		Ende der Programmausführung

Codierung der Operanden

Register sind durchnummeriert (von 0 bis 31). Der *Program Counter* steht stets in Register 31. Für Rücksprungadressen wird üblicherweise Register 30 verwendet. Der Top-of-Stack-Pointer wird in Register 29 abgelegt.

Immediate-Operanden werden immer im Zweierkomplement der für sie zur Verfügung stehenden Teilwortbreite codiert. (Außer bei den Größenvergleichen hat dies allerdings keine Auswirkungen.)

Der Speicher ist wortweise adressiert, das heißt, Speicheradresse 0 bezeichnet das erste Speicherwort, Speicheradresse 1 das zweite usw.; also bedeutet Speicheradresse 0 das erste Byte, Speicheradresse 1 das fünfte usw.