The Haverford Educational RISC Architecture (HERA) Quick Guide for Version 2.4

Registers and Flags

16-bit registers: R_0: zero; R_1...R_{10}: general; R_{11}....R_{13}: idiomatic; R_{14}: frame ptr.; R_{15}: stack ptr.
1-bit flags: F_0: sign; F_1: zero; F_2: overflow; F_3: carry; F_4: carry-block

Arithmetic/Logical/Shift Instructions

Mnemonic	Meaning	Op. Code	Notes
SETLO(d, v)	$R_d \leftarrow v$	`E` d v v	set R_d to signed quantity v (8-bits)
SETHI(d, v)	$(R_d)_{15:8} \leftarrow v$	`F` d v v	set high 8 bits of R_d
AND(d, a, b)	$R_d(i) \leftarrow R_a(i) \wedge R_b(i)$	`8` d a b	bit-wise logical and
OR(d, a, b)	$R_d(i) \leftarrow R_a(i) \vee R_b(i)$	`9` d a b	bit-wise logical or
XOR(d, a, b)	$R_d(i) \leftarrow R_a(i) \oplus R_b(i)$	`D` d a b	bit-wise logical exclusive or
ADD(d, a, b)	$R_d \leftarrow R_a + R_b + (c \wedge F_4')$	`A` d a b	addition, use carry unless blocked
SUB(d, a, b)	$R_d \leftarrow R_a - R_b - (c' \wedge F_4')$	`B` d a b	subtraction, use carry unless blocked
MUL(d, a, b)	$R_d \leftarrow (R_a * R_b)_{15:0\,[\text{Or}\,31:16]}$	`C` d a b	multiplication: low bits if $F_4 = 1$...
INC(d, δ)	$R_d \leftarrow R_d + \delta$	`3` d `10`$\epsilon\epsilon$ $\epsilon\epsilon\epsilon\epsilon$	increment R_d by δ (where $\epsilon = \delta - 1$)
DEC(d, δ)	$R_d \leftarrow R_d - \delta$	`3` d `11`$\epsilon\epsilon$ $\epsilon\epsilon\epsilon\epsilon$	decrement R_d by δ (where $\epsilon = \delta - 1$)
LSL(d, b)	$R_d \leftarrow$ shl/rolc (R_b)	`3` d `0` b	logical shift left, possibly with carry
LSR(d, b)	$R_d \leftarrow$ shr/rorc (R_b)	`3` d `1` b	logical shift right, possibly with carry
LSL8(d, b)	$R_d \leftarrow$ shl8 (R_b)	`3` d `2` b	logical shift left 8 bits
LSR8(d, b)	$R_d \leftarrow$ shr8 (R_b)	`3` d `3` b	logical shift right 8 bits
ASL(d, b)	$R_d \leftarrow$ asl/aslc (R_b)	`3` d `4` b	arithmetic shift left, possibly with carry
ASR(d, b)	$R_d \leftarrow$ asr (R_b)	`3` d `5` b	arithmetic shift right

Flag Manipulation Instructions

Mnemonic	Meaning	Op. Code	Notes
FON(v)	$F \leftarrow F \vee v$	`3 000`v `6` $vvvv$	Set to true any flags for which v is true
FOFF(v)	$F \leftarrow F \wedge v'$	`3 100`v `6` $vvvv$	Set to false any flags for which v is true
FSET5(v)	$F \leftarrow v$	`3 010`v `6` $vvvv$	Set all flags to have the values v
FSET4(v)	$F_{3:0} \leftarrow v$	`3 1100 6` $vvvv$	Set flags other than carry-block
SAVEF(d)	$R_d \leftarrow F$	`3` d `7 0`	Save flags to R_d
RSTRF(d)	$F \leftarrow R_d$	`3` d `7 8`	Restore flags from R_d

Memory Access Instructions

Mnemonic	Meaning	Op. Code	Notes
LOAD(d, o, b)	$R_d \leftarrow M[R_b + o]$	`010`o d $oooo$ b	load from $R_b + o$ (o is 5-bit *unsigned*)
STORE(d, o, b)	$M[R_b + o] \leftarrow R_d$	`011`o d $oooo$ b	store to $R_b + o$ (o is 5-bit *unsigned*)

Branch Instructions (see Mano, Ch. 9-8)

Mnemonic	Meaning	Op. Code	Notes
BR(b)	$PC \leftarrow R_b$	`1 0 0` b	Unconditional branch – *true*
BL(b)	$PC \leftarrow R_b$ if $(s \oplus v)$	`1 2 0` b	Branch if signed result <0
BGE(b)	$PC \leftarrow R_b$ if $(s \oplus v)'$	`1 3 0` b	Branch if signed result $\geqslant 0$
BLE(b)	$PC \leftarrow R_b$ if $((s \oplus v) \vee z)$	`1 4 0` b	Branch if signed result $\leqslant 0$
BG(b)	$PC \leftarrow R_b$ if $((s \oplus v) \vee z)'$	`1 5 0` b	Branch if signed result >0
BULE(b)	$PC \leftarrow R_b$ if $(c' \vee z)$	`1 6 0` b	Branch if unsigned result $\leqslant 0$
BUG(b)	$PC \leftarrow R_b$ if $(c' \vee z)'$	`1 7 0` b	Branch if unsigned result >0
BZ(b)	$PC \leftarrow R_b$ if z	`1 8 0` b	Branch if zero/if equal
BNZ(b)	$PC \leftarrow R_b$ if z'	`1 9 0` b	Branch if not zero/not equal
BC(b)	$PC \leftarrow R_b$ if c	`1 A 0` b	Branch if carry/unsigned result $\geqslant 0$
BNC(b)	$PC \leftarrow R_b$ if c'	`1 B 0` b	Branch if not carry/unsigned result <0
BS(b)	$PC \leftarrow R_b$ if s	`1 C 0` b	Branch if sign (negative)
BNS(b)	$PC \leftarrow R_b$ if s'	`1 D 0` b	Branch if not sign (non-negative)
BV(b)	$PC \leftarrow R_b$ if v	`1 E 0` b	Branch if overflow
BNV(b)	$PC \leftarrow R_b$ if v'	`1 F 0` b	Branch if not overflow
BRR(o)	$PC \leftarrow PC + o$	`0 0` o o	Relative branch by o (o is 8-bit *signed*)
...			(All branches can also be relative)

Function/Interrupt Instructions

Mnemonic	Meaning	Op	Notes
CALL(a,b)	$PC \leftarrow R_b$, $R_b \leftarrow PC+1$, $FP \leftarrow R_a, R_a \leftarrow FP$	`2 0` a b	Call function at address R_b, with stack at R_a (or, equivalently, return or co-routine switch)
RETURN(a,b)	$PC \leftarrow R_b$, $R_b \leftarrow PC+1$, $FP \leftarrow R_a, R_a \leftarrow FP$	`2 1` a b	Return from function, expecting return address in R_b and caller's FP in R_a
SWI(i)		`2 2 0` i	Software interrupt $\#i$
RTI()		`2 3 0 0`	Return from interrupt

Pseudo-Instructions and Data Statements

Mnemonic	Definition	Notes
SET(d, v)	SETLO(d, v `&` 0$x$ff); SETHI(d, $v \gg 8$)	$R_d \leftarrow v$ (set R_d to 16-bit value v)
SETRF(d, v)	SET(d, l); FLAGS(d)	$R_d \leftarrow v$ (set R_d **and flags** for $v+0$)
MOVE(a, b)	OR(a, b,R_0)	$R_a \leftarrow R_b$
CMP(a, b)	CON( ); SUB(R_0, a, b)	Set flags for $R_a - R_b$
NEG(d, b)	CON( ); SUB(d, R_0, b)	Set $R_d \leftarrow -R_b$
NOT(d, b)	SET(R_t, 0x ffff); XOR(d, R_t, b)	Bitwise complement
CON( )	FON(0x08)	Turn on the carry flag (no borrow)
COFF( )	FOFF(0x08)	Turn off the carry flag (no carry)
CBON( )	FON(0x10)	Turn on the carry-block flag
CCBOFF( )	FOFF(0x18)	Turn off carry and carry-block flags
FLAGS(a)	COFF( ); ADD(R_0, R_a, R_0)	Set flags for R_a
LABEL(L)/DLABEL(L)	*(no machine language generated)*	Define a label or data label L
INTEGER(i)	i	Put i in the current memory cell
LP_STRING(s)	s	Put string s in memory for Tiger
DSKIP(n)	n uninitialized data memory cells	Skip n cells of data memory
CONSTANT(N, v)	*(no machine language generated)*	Define name N to have value v
HALT( )	BRR(0)	Halt the program
NOP( )	BRR(1)	Do nothing ("No operation")
OPCODE(n)	n	Machine Language op n