HCS12 Instruction Reference

Instr.	Description
ABA	add accumulator B to accumulator A
ABX	add accumulator B to index reg. X
ABY	add accumulator B to index reg. Y
ADCA *src*	add with carry to A
ADCB *src*	add with carry to B
ADDA *src*	add to accumulator A
ADDB *src*	add to accumulator B
ADDD *src*	add 16-bit to D
ANDA *src*	logical AND A with memory
ANDB *src*	logical AND B with memory
ANDCC *#mask*	logical and CCR with mask
ASL *src*	arithmetic shift left memory
ASLA	arithmetic shift left A
ASLB	arithmetic shift left B
ASLD	arithmetic shift left D
ASR *src*	arithmetic shift right memory
ASRA	arithmetic shift right A
ASRB	arithmetic shift right B
BCC *dest*	branch if carry clear
BCLR *dest,mask*	clear bit(s) in memory
BCS *dest*	branch if carry set
BEQ *dest*	branch if equal
BGE *dest*	branch if >= zero
BGND	enter background debug mode
BGT *dest*	branch if > zero
BHI *dest*	branch if higher
BHS *dest*	branch if higher or same
BITA *src*	bit test A with memory
BITB *src*	bit test B with memory
BLE *dest*	branch if <= zero
BLO *dest*	branch if lower
BLS *dest*	branch if lower or same
BLT *dest*	branch if < zero
BMI *dest*	branch if minus
BNE *dest*	branch if not equal to zero
BPL *dest*	branch if plus
BRA *dest*	branch always
BRCLR *src,mask,dest*	branch if bit(s) clear
BRN *dest*	branch never

Instr.	Description
BRSET *src,mask,dest*	branch if bit(s) set
BSET *dest,mask*	set bit(s) in memory
BSR *dest*	branch to subroutine
BVC *dest*	branch if overflow clear
BVS *dest*	branch if overflow set
CALL *dest*	call subroutine in extended memory
CBA	compare accumulators A & B
CLC	clear carry
CLI	clear interrupt mask
CLR *dest*	clear memory
CLRA	clear A
CLRB	clear B
CLV	clear two's complement overflow bit
CMPA *src*	compare A with memory
CMPB *src*	compare B with memory
COM *dest*	complement memory
COMA *src*	complement A
COMB *src*	complement B
CPD *src*	compare D with memory
CPS *src*	compare stack pointer
CPX *src*	compare index reg. X
CPY *src*	compare index reg. Y
DAA	decimal adjust A
DBEQ *reg,dest*	decr. and branch if equal to zero
DBNE *reg,dest*	decr. and branch if not zero
DEC *dest*	decrement memory
DECA	decrement A
DECB	decrement B
DES	decrement stack pointer
DEX	decrement index register X
DEY	decrement index register Y
EDIV	Extn'd divide 32 by 16-bit (unsigned)
EDIVS	extended divide 32 by 16-bit (signed)
EMACS *src*	Extn'd mult. and accumulate (signed)
EMAXD *src*	max of 2 16-bit values (to D)
EMAXM *dest*	max of 2 16-bit values (to mem)
EMIND *src*	min of 2 16-bit values (to D)
EMINM *dest*	min of 2 16-bit values (to mem)
EMUL	16 by 16-bit multiply (unsigned)

Instr.	Description
EMULS	16 by 16-bit multiply (signed)
EORA *src*	exclusive-OR A with memory
EORB *src*	exclusive-OR B with memory
ETBL *src*	16-bit table lookup and interpolate
EXG *reg,reg*	exchange register contents
FDIV	16 by 16-bit fractional divide
IBEQ *reg,dest*	incr. and branch if equal to zero
IBNE *reg,dest*	incr. and branch if not equal to zero
IDIV	16 by 16-bit integer divide
IDIVS	16 by 16-bit integer divide (signed)
INC *dest*	increment memory
INCA	increment A
INCB	increment B
INS	increment stack pointer
INX	increment index register X
INY	increment index register Y
JMP *dest*	jump
JSR *dest*	jump to subroutine
LBCC *dest*	long branch if carry clear
LBCS *dest*	long branch if carry set
LBEQ *dest*	long branch if equal
LBGE *dest*	long branch if >= zero
LBGT *dest*	long branch if greater than zero
LBHI *dest*	long branch if higher
LBHS *dest*	long branch if higher or same
LBLE *dest*	long branch if <= zero
LBLO *dest*	long branch if lower
LBLS *dest*	long branch if lower or same
LBLT *dest*	long branch if less than zero
LBMI *dest*	long branch if minus
LBNE *dest*	long branch if not equal to zero
LBPL *dest*	long branch if plus
LBRA *dest*	long branch always
LBRN *dest*	long branch never
LBVC *dest*	long branch if overflow clear
LBVS *dest*	long branch if overflow set
LDAA *src*	load accumulator A
LDAB *src*	load accumulator B
LDD *src*	load accumulator D

For more information on instructions see the S12CPUV2 Reference Manual

HCS12 Instruction Reference

Instr.	Description
LDS *src*	load stack pointer
LDX *src*	load index register X
LDY *src*	load index register Y
LEAS *src*	load effective address into SP
LEAX *src*	load effective address into X
LEAY *src*	load effective address into Y
LSL *dest*	logical shift left memory
LSLA	logical shift left A
LSLB	logical shift left B
LSLD	logical shift left D
LSR *dest*	logical shift right memory
LSRA	logical shift right A
LSRB	logical shift right B
LSRD	logical shift right D
MAXA *src*	max of 2 8-bit values (to A)
MAXM *dest*	max of 2 8-bit values (to mem)
MEM	determine grade of membership
MINA *src*	min of 2 8-bit values (to A)
MINM *dest*	min of 2 8-bit values (to mem)
MOVB *src,dest*	memory to memory byte move
MOVW *src,dest*	memory to memory word move
MUL	8 by 8-bit multiply (unsigned)
NEG *dest*	negate (2'a complement) memory
NEGA	negate A
NEGB	negate B
NOP	no operation
ORAA *src*	inclusive OR A with memory
ORAB *src*	inclusive OR B with memory
ORCC *#mask*	logical OR CCR with mask
PSHA	push A onto stack
PSHB	push B onto stack

Instr.	Description
PSHC	push CCR onto stack
PSHD	push D onto stack
PSHX	push index reg. X onto stack
PSHY	push index reg. Y onto stack
PULA	pull A from stack
PULB	pull B from stack
PULC	pull CCR from stack
PULD	pull D from stack
PULX	pull index reg. X from stack
PULY	pull index reg. Y from stack
REV	fuzzy logic rule evaluation
REVW	fuzzy logic rule evaluation (weighted)
ROL *dest*	rotate left memory
ROLA	rotate left A
ROLB	rotate left B
ROR *dest*	rotate right memory
RORA	rotate right A
RORB	rotate right B
RTC	return from call
RTI	return from interrupt
RTS	return from subroutine
SBA	subtract accumulator B from A
SBCA *src*	subtract with borrow from A
SBCB *src*	subtract with borrow from B
SEC	set carry
SEI	set interrupt mask
SEV	set two's complement overflow bit
SEX *reg8,reg16*	sign extend into 16-bit register
STAA *dest*	store A to memory
STAB *dest*	store B to memory
STD *dest*	store D to memory

Instr.	Description
STOP	stop processing
STS *dest*	store stack pointer
STX *dest*	store index register X
STY *dest*	store index register Y
SUBA *src*	subtract memory from A
SUBB *src*	subtract memory from B
SUBD *src*	subtract memory from D
SWI	software interrupt
TAB	transfer A to B
TAP	transfer A to CCR
TBA	transfer B to A
TBEQ *reg,dest*	test and branch if equal to zero
TBL *src*	8-bit table lookup and interpolate
TBNE *reg,dest*	test and branch if not equal to zero
TFR *reg,reg*	transfer register to another register
TPA	transfer CCR to A
TRAP	unimplemented opcode trap
TST *src*	test memory
TSTA	test A
TSTB	test B
TSX	transfer SP to index reg. X
TSY	transfer SP to index reg. Y
TXS	transfer index reg. X to SP
TYS	transfer index reg. Y to SP
WAI	wait for interrupts
WAV	weighted average
WAVR	resume WAV
XGDX	exchange D and index reg. X
XGDY	exchange D and index reg. Y

Note: Operand field in instruction:

src	- operand is source of data
dest	- operand is destination
reg	- operand is a register
reg8	- operand is an 8-bit register
reg16	- operand is a 16-bit register
mask	- operand is a constant used as a mask
blank	- inherent or special addressing mode

For more information on instructions see the S12CPUV2 Reference Manual

Version 2 – September 17, 2010, Paul Michaud