XILINX®

Chapter 3

PicoBlaze Instruction Set

Table 3-1 summarizes the entire PicoBlaze instruction set, which appears alphabetically. Instructions are listed using the KCPSM3 syntax. If different, the pBlazIDE syntax appears in parentheses. Each instruction includes an overview description, a functional description, and how the ZERO and CARRY flags are affected. For more details on each instruction, see Appendix C, "PicoBlaze Instruction Set and Event Reference."

Table 3-1: **PicoBlaze Instruction Set (alphabetical listing)**

Instruction	Description	Function	ZERO	CARRY
ADD sX, kk	Add register sX with literal kk	sX ← sX + kk	?	?
ADD sX, sY	Add register sX with register sY	sX ← sX + sY	?	?
ADDCY sX, kk (ADDC)	Add register sX with literal kk with CARRY bit	sX ← sX + kk + CARRY	?	?
ADDCY sX, sY (ADDC)	Add register sX with register sY with CARRY bit	sX ← sX + sY + CARRY	?	?
AND sX, kk	Bitwise AND register sX with literal kk	sX ← sX AND kk	?	0
AND sX, sY	Bitwise AND register sX with register sY	sX ← sX AND sY	?	0
CALL aaa	Unconditionally call subroutine at aaa	TOS ← PC PC ← aaa	-	-
CALL C, aaa	If CARRY flag set, call subroutine at aaa	If CARRY=1, {TOS ← PC, PC ← aaa}	-	-
CALL NC, aaa	If CARRY flag not set, call subroutine at aaa	If CARRY=0, {TOS ← PC, PC ← aaa}	-	-
CALL NZ, aaa	If ZERO flag not set, call subroutine at aaa	If ZERO=0, {TOS ← PC, PC ← aaa}	-	-
CALL Z, aaa	If ZERO flag set, call subroutine at aaa	If ZERO=1, {TOS ← PC, PC ← aaa}	-	-
COMPARE sX, kk (COMP)	Compare register sX with literal kk. Set CARRY and ZERO flags as appropriate. Registers are unaffected.	If sX=kk, ZERO ← 1 If sX<kk, CARRY ← 1	?	?
COMPARE sX, sY (COMP)	Compare register sX with register sY. Set CARRY and ZERO flags as appropriate. Registers are unaffected.	If sX=sY, ZERO ← 1 If sX<sY, CARRY ← 1	?	?
DISABLE INTERRUPT (DINT)	Disable interrupt input	INTERRUPT_ENABLE ← 0	-	-

Table 3-1: **PicoBlaze Instruction Set (alphabetical listing)**

Instruction	Description	Function	ZERO	CARRY
ENABLE INTERRUPT (EINT)	Enable interrupt input	INTERRUPT_ENABLE ← 1	-	-
Interrupt Event	Asynchronous interrupt input. Preserve flags and PC. Clear INTERRUPT_ENABLE flag. Jump to interrupt vector at address 3FF.	Preserved ZERO ← ZERO Preserved CARRY ← CARRY INTERRUPT_ENABLE ← 0 TOS ← PC PC ← 3FF	-	-
FETCH sX, (sY) (FETCH sX, sY)	Read scratchpad RAM location pointed to by register `sY` into register `sX`	sX ← RAM[(sY)]	-	-
FETCH sX, ss	Read scratchpad RAM location ss into register `sX`	sX ← RAM[ss]	-	-
INPUT sX, (sY) (IN sX, sY)	Read value on input port location pointed to by register `sY` into register `sX`	PORT_ID ← sY sX ← IN_PORT	-	-
INPUT sX, pp (IN)	Read value on input port location pp into register `sX`	PORT_ID ← pp sX ← IN_PORT	-	-
JUMP aaa	Unconditionally jump to aaa	PC ← aaa	-	-
JUMP C, aaa	If CARRY flag set, jump to aaa	If CARRY=1, PC ← aaa	-	-
JUMP NC, aaa	If CARRY flag not set, jump to aaa	If CARRY=0, PC ← aaa	-	-
JUMP NZ, aaa	If ZERO flag not set, jump to aaa	If ZERO=0, PC ← aaa	-	-
JUMP Z, aaa	If ZERO flag set, jump to aaa	If ZERO=1, PC ← aaa	-	-
LOAD sX, kk	Load register `sX` with literal `kk`	sX ← kk	-	-
LOAD sX, sY	Load register `sX` with register `sY`	sX ← sY	-	-
OR sX, kk	Bitwise OR register `sX` with literal `kk`	sX ← sX OR kk	?	0
OR sX, sY	Bitwise OR register `sX` with register `sY`	sX ← sX OR sY	?	0
OUTPUT sX, (sY) (OUT sX, sY)	Write register `sX` to output port location pointed to by register `sY`	PORT_ID ← sY OUT_PORT ← sX	-	-
OUTPUT sX, pp (OUT sX, pp)	Write register `sX` to output port location pp	PORT_ID ← pp OUT_PORT ← sX	-	-
RETURN (RET)	Unconditionally return from subroutine	PC ← TOS+1	-	-
RETURN C (RET C)	If CARRY flag set, return from subroutine	If CARRY=1, PC ← TOS+1	-	-
RETURN NC (RET NC)	If CARRY flag not set, return from subroutine	If CARRY=0, PC ← TOS+1	-	-
RETURN NZ (RET NZ)	If ZERO flag not set, return from subroutine	If ZERO=0, PC ← TOS+1	-	-
RETURN Z (RET Z)	If ZERO flag set, return from subroutine	If ZERO=1, PC ← TOS+1	-	-

Table 3-1: **PicoBlaze Instruction Set (alphabetical listing)**

Instruction	Description	Function	ZERO	CARRY
RETURNI DISABLE (RETI DISABLE)	Return from interrupt service routine. Interrupt remains disabled.	PC ← TOS ZERO ← Preserved ZERO CARRY ← Preserved CARRY INTERRUPT_ENABLE ← 0	?	?
RETURNI ENABLE (RETI ENABLE)	Return from interrupt service routine. Re-enable interrupt.	PC ← TOS ZERO ← Preserved ZERO CARRY ← Preserved CARRY INTERRUPT_ENABLE ← 1	?	?
RL sX	Rotate register sX left	sX ← {sX[6:0],sX[7]} CARRY ← sX[7]	?	?
RR sX	Rotate register sX right	sX ← {sX[0],sX[7:1]} CARRY ← sX[0]	?	?
SL0 sX	Shift register sX left, zero fill	sX ← {sX[6:0],0} CARRY ← sX[7]	?	?
SL1 sX	Shift register sX left, one fill	sX ← {sX[6:0],1} CARRY ← sX[7]	0	?
SLA sX	Shift register sX left through all bits, including CARRY	sX ← {sX[6:0],CARRY} CARRY ← sX[7]	?	?
SLX sX	Shift register sX left. Bit sX[0] is unaffected.	sX ← {sX[6:0],sX[0]} CARRY ← sX[7]	?	?
SR0 sX	Shift register sX right, zero fill	sX ← {0,sX[7:1]} CARRY ← sX[0]	?	?
SR1 sX	Shift register sX right, one fill	sX ← {1,sX[7:1]} CARRY ← sX[0]	0	?
SRA sX	Shift register sX right through all bits, including CARRY	sX ← {CARRY,sX[7:1]} CARRY ← sX[0]	?	?
SRX sX	Arithmetic shift register sX right. Sign extend sX. Bit sX[7] Is unaffected.	sX ← {sX[7],sX[7:1]} CARRY ← sX[0]	?	?
STORE sX, (sY) (STORE sX, sY)	Write register sX to scratchpad RAM location pointed to by register sY	RAM[(sY)] ← sX	-	-
STORE sX, ss	Write register sX to scratchpad RAM location ss	RAM[ss] ← sX	-	-
SUB sX, kk	Subtract literal kk from register sX	sX ← sX – kk	?	?
SUB sX, sY	Subtract register sY from register sX	sX ← sX – sY	?	?
SUBCY sX, kk (SUBC)	Subtract literal kk from register sX with CARRY (borrow)	sX ← sX – kk - CARRY	?	?
SUBCY sX, sY (SUBC)	Subtract register sY from register sX with CARRY (borrow)	sX ← sX – sY - CARRY	?	?

Table 3-1: **PicoBlaze Instruction Set (alphabetical listing)**

Instruction	Description	Function	ZERO	CARRY
TEST sX, kk	Test bits in register sX against literal kk. Update CARRY and ZERO flags. Registers are unaffected.	If (sX AND kk) = 0, ZERO ← 1 CARRY ← odd parity of (sX AND kk)	?	?
TEST sX, sY	Test bits in register sX against register sX. Update CARRY and ZERO flags. Registers are unaffected.	If (sX AND sY) = 0, ZERO ← 1 CARRY ← odd parity of (sX AND kk)	?	?
XOR sX, kk	Bitwise XOR register sX with literal kk	sX ← sX XOR kk	?	0
XOR sX, sY	Bitwise XOR register sX with register sY	sX ← sX XOR sY	?	0

sX = One of 16 possible register locations ranging from s0 through sF or specified as a literal

sY = One of 16 possible register locations ranging from s0 through sF or specified as a literal

aaa = 10-bit address, specified either as a literal or a three-digit hexadecimal value ranging from 000 to 3F or a labeled location

kk = 8-bit immediate constant, specified either as a literal or a two-digit hexadecimal value ranging from 00 to FF or specified as a literal

pp = 8-bit port address, specified either as a literal or a two-digit hexadecimal value ranging from 00 to FF or specified as a literal

ss = 6-bit scratchpad RAM address, specified either as a literal or a two-digit hexadecimal value ranging from 00 to 3F or specified as a literal

RAM[n] = Contents of scratchpad RAM at location n

TOS = Value stored at Top Of Stack

Address Spaces

As shown in Table 3-2, the PicoBlaze microcontroller has five distinct address spaces. Specific instructions operate on each of the address spaces.

Table 3-2: **PicoBlaze Address Spaces and Related Instructions**

Address Space	Size (Depth x Width)	Addressing Modes	Instructions that Operate on Address Space
Instruction	1Kx18	Direct	• JUMP • CALL • RETURN • RETURNI • INTERRUPT event • RESET event All others increment the PC to the next location
Register File	16x8	Direct	• LOAD • AND • OR • XOR • TEST (read only) • ADD • ADDCY • SUB • SUBCY • COMPARE (read only) • SR0 • SR1 • SRX • SRA • RR • SL0 • SL1 • SLX • SLA • RL • INPUT • OUTPUT (read only) • STORE (read only) • FETCH
Scratchpad RAM	64x8	Direct Indirect	• STORE • FETCH
I/O	256x8	Direct Indirect	• INPUT • OUTPUT
CALL/RETURN Stack	31x10	N/A	• CALL • Enabled INTERRUPT event • RETURN • RETURNI • RESET event