Arithmetic Operations

	Short Description:	Definition:	Full Description:
ADD	Unsigned integer add	ADD REG1, REG2, OP(255)	Performs 32-bit add on two 32-bit zero extended source values
ADC	Unsigned integer add (carry)	ADC REG1, REG2, OP(255)	Performs 32-bit add on two 32-bit zero extended source values, plus a stored carry bit
SUB	Unsigned integer subtract	SUB REG1, REG2, OP(255)	Performs 32-bit subtract on two 32-bit zero extended source values
SUC	Unsigned integer subtract (carry)	SUC REG1, REG2, OP(255)	Performs 32-bit subtract on two 32-bit zero extended source values with carry (borrow)
RSB	Reverse unsigned int subtract	RSB REG1, REG2, OP(255)	Performs 32-bit subtract on two 32-bit zero extended source values. Source values reversed
RSC	Reverse unsigned integer subtract (carry)	RSC REG1, REG2, OP(255)	Performs 32-bit subtract on two 32-bit zero extended source values with carry (borrow). Source values reversed

Logical Operations

LSL	Logical shift left	LSL REG1, REG2, OP(31)	Performs 32-bit shift left of the zero extended source value
LSR	Logical shift right	LSR REG1, REG2, OP(31)	Performs 32-bit shift right of the zero extended source value
AND	Bitwise AND	AND REG1, REG2, OP(255)	Performs 32-bit logical AND on two 32-bit zero extended source values
OR	Bitwise OR	OR REG1, REG2, OP(255)	Performs 32-bit logical OR on two 32-bit zero extended source values
XOR	Bitwise XOR	XOR REG1, REG2, OP(255)	Performs 32-bit logical XOR on two 32-bit zero extended source values
NOT	Bitwise NOT	NOT REG1, REG2	Performs 32-bit logical NOT on the 32-bit zero extended source value
MIN	Copy minimum	MIN REG1, REG2, OP(255)	Compares two 32-bit zero extended source values and copies the smaller to REG1
MAX	Copy maximum	MAX REG1, REG2, OP(255)	Compares two 32-bit zero extended source values and copies the larger to REG1
CLR	Clear bit	CLR REG1, REG2, OP(31)	Clears the specified bit in the source and copies the result to the destination Also: **CLR REG1, OP(31)** **CLR REG1, Rn.tx** **CLR Rn.tx**
SET	Set bit	SET REG1, REG2, OP(31)	Sets the specified bit in the source and copies the result to the destination Also: **SET REG1, OP(31)** **SET REG1, Rn.tx** **SET Rn.tx**
SCAN	Register field scan	SCAN Rn, OP(255)	The SCAN instruction scans the register file for a particular value. It includes a configurable field width and stride. The width of the field to match can be set to 1, 2, or 4 bytes.
LMBD	Left-most bit detect	LMBD REG1, REG2, OP(255)	Scans REG2 from its left-most bit for a bit value matching bit 0 of OP(255), and writes the bit number in REG1 (writes 32 to REG1 if the bit is not found)

Register Load and Store

MOV	Copy value	MOV REG1, OP(65535)	Moves the value from OP(65535), zero extends it, and stores it into REG1
LDI	Load immediate	LDI REG1, IM(65535)	The LDI instruction moves value from IM(65535), zero extends it, and stores it into REG1
MVIB	Move register file indirect (8)	MVIB [*&]REG1, [*&]REG2	The MVIx instruction family moves a value from the source to the destination. The source, destination, or both can be register pointers.
MVIW	Move register file indirect (16)	MVIW [*&]REG1, [*&]REG2	
MVID	Move register file indirect (32)	VID [*&]REG1, [*&]REG2	
LBBO	Load byte burst	LBBO REG1, Rn2, OP(255), IM(124) LBBO REG1, Rn2, OP(255), bn	The LBBO instruction is used to read a block of data from memory into the register file. The memory address to read from is specified by a 32-bit register, using an optional offset
SBBO	Store byte burst	SBBO REG1, Rn2, OP(255), IM(124) SBBO REG1, Rn2, OP(255), bn	The SBBO instruction is used to write a block of data from the register file into memory. The memory address to which to write is specified by a 32-bit register, using an optional offset
LBCO	Load byte burst with constant table offset	LBCO REG1, Cn2, OP(255), IM(124) LBCO REG1, Cn2, OP(255), bn	The LBCO instruction is used to read a block of data from memory into the register file. The memory address from which to read is specified by a 32-bit constant register (Cn2), using an optional offset from an immediate or register value
SBCO	Store byte burst with constant table offset	SBCO REG1, Cn2, OP(255), IM(124) SBCO REG1, Cn2, OP(255), bn	The SBCO instruction is used to write a block of data from the register file into memory. The memory address to write to is specified by a 32-bit constant register (Cn2), using an optional offset from an immediate or register value
ZERO	Clear register space	ZERO IM(123), IM(124)	This pseudo-op is used to clear space in the register file. Also: **ZERO &REG1, IM(124)**

Program Flow Control

JMP	Unconditional jump	JMP OP(65535)	Unconditional jump to a 16-bit instruction address, specified by register or immediate value
JAL	Unconditional jump and link	JAL REG1, OP(65535)	Unconditional jump to a 16-bit instruction address, specified by register or immediate value. Address following the JAL instruction is stored into REG1, so that REG1 can later be used as a "return" address
CALL	Call procedure	CALL OP(65535)	The CALL instruction is designed to emulate a subroutine call on a stack-based processor
RET	Return from procedure	RET	The RET instruction is designed to emulate a subroutine return on a stack-based processor
QBGT	Quick branch if >	QBGT LABEL, REG1, OP(255)	Jumps if the value of OP(255) is greater than REG1
QBGE	Quick branch if ≥	QBGE LABEL, REG1, OP(255)	Jumps if the value of OP(255) is greater than or equal to REG1
QBLT	Quick branch if <	QBLT LABEL, REG1, OP(255)	Jumps if the value of OP(255) is less than REG1
QBLE	Quick branch if ≤	QBLE LABEL, REG1, OP(255)	Jumps if the value of OP(255) is less than or equal to REG1
QBEQ	Quick branch if =	QBEQ LABEL, REG1, OP(255)	Jumps if the value of OP(255) is equal to REG1
QBNE	Quick branch if ≠	QBNE LABEL, REG1, OP(255)	Jumps if the value of OP(255) is NOT equal to REG1
QBA	Quick branch always	QBA LABEL	Jump always. This is similar to the JMP instruction, only QBA uses an address offset and thus can be relocated in memory
QBBS	Quick branch if bit is set	QBBS LABEL, REG1, OP(31)	Jumps if the bit OP(31) is set in REG1. Also: **QBBS LABEL, Rn.tx**
QBBC	Quick branch if bit is clear	QBBC LABEL, REG1, OP(31)	Jumps if the bit OP(31) is clear in REG1. Also: **QBBC LABEL, Rn.tx**
WBS	Wait until bit set	WBS REG1, OP(31) WBS Rn.tx	The WBS instruction is a pseudo op that uses the QBBC instruction. It is used to poll on a status bit, spinning until the bit is set
WBC	Wait until bit clear	WBC REG1, OP(31) WBC Rn.tx	The WBC instruction is a pseudo op that uses the QBBS instruction. It is used to poll on a status bit, spinning until the bit is clear
HALT	Halt operation	HALT	The HALT instruction disables the PRU. This instruction is used to implement software breakpoints in a debugger
SLP	Sleep operation	SLP IM(1)	The SLP instruction will sleep the PRU, causing it to disable its clock. This instruction can specify either a permanent sleep or a "wake on event"

See: **http://processors.wiki.ti.com/index.php/PRU_Assembly_Instructions** for further information.

REG, REG1, REG2, ...	A register field from 8 to 32 bits	bn	A field that must be b0 to b3
Rn, Rn1, Rn2, ...	A 32-bit register field (r0 to r31)	LABEL	A valid label
Rn.tx	A 1-bit register field	IM(n)	An immediate value from 0 to n
Cn, Cn1, Cn2, ...	A 32-bit constant constant register (c0 to c31)	OP(n)	Operand - either a REG or IM(n)

EXPLORING**BEAGLEBONE**
TOOLS AND TECHNIQUES FOR BUILDING WITH EMBEDDED LINUX

Summary of the PRU instruction set

www.ExploringBeagleBone.com