ECOLE SUPERIEURE
DE TECHNOLOGIE ELECTRONIQUE

CITÉ DESCARTES - BP 99
93162 NOISY-LE-GRAND CEDEX
TÉL. : 01 45 92 65 00 - FAX : 01 45 92 66 99
INTERNET : http://www.esiee.fr

Instructions By Category

Année scolaire : 2003-2004

CHAMBRE DE COMMERCE ET D'INDUSTRIE DE PARIS

Table B-5. Storage Reference Instructions

Mnemonic	Operands	Function	Other Registers Changed
lbz	RT, D(RA)	Load byte from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) ← $^{24}0$ ‖ MS(EA,1).	
lhz	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) ← $^{16}0$ ‖ MS(EA,2).	
lwz	RT, D(RA)	Load word from EA = (RAI0) + EXTS(D) and place in RT, (RT) ← MS(EA,4).	
stb	RS, D(RA)	Store byte $(RS)_{24:31}$ in memory at EA = (RAI0) + EXTS(D).	
sth	RS, D(RA)	Store halfword $(RS)_{16:31}$ in memory at EA = (RAI0) + EXTS(D).	
stw	RS, D(RA)	Store word (RS) in memory at EA = (RAI0) + EXTS(D).	
li	RT, IM	Load immediate. (RT) ← EXTS(IM) *Extended mnemonic* for addi RT,0,value	
lis	RT, IM	Load immediate shifted. (RT) ← (IM ‖ $^{16}0$) *Extended mnemonic* for addis RT,0,value	
mfdcr	RT, DCRN	Move from DCR to RT, (RT) ← (DCR(DCRN)).	
mtdcr	DCRN, RS	Move to DCR from RS, (DCR(DCRN)) ← (RS).	

Table B-8. Branch Instructions

Mnemonic	Operands	Function	Other Registers Changed
b	target	Branch unconditional relative. LI ← (target – CIA)$_{6:29}$ NIA ← CIA + EXTS(LI ‖ $^{2}0$)	
ba		Branch unconditional absolute. LI ← target$_{6:29}$ NIA ← EXTS(LI ‖ $^{2}0$)	
bl		Branch unconditional relative. LI ← (target – CIA)$_{6:29}$ NIA ← CIA + EXTS(LI ‖ $^{2}0$)	(LR) ← CIA + 4.
beq	[cr_field,] target	Branch if equal. Use CR0 if cr_field is omitted. *Extended mnemonic* for bc 12,4*cr_field+2,target	
bne	[cr_field,] target	Branch if not equal. Use CR0 if cr_field is omitted. *Extended mnemonic* for bc 4,4*cr_field+2,target	

Table B-6. Arithmetic and Logical Instructions

Mnemonic	Operands	Function	Other Registers Changed
add	RT, RA, RB	Add (RA) to (RB). Place result in RT.	
add.			CR[CR0]
addis	RT, RA, IM	Add (IM ‖ $^{16}0$) to (RA|0). Place result in RT.	
and	RA, RS, RB	AND (RS) with (RB). Place result in RA.	
and.			CR[CR0]
andi.	RA, RS, IM	AND (RS) with ($^{16}0$ ‖ IM). Place result in RA.	CR[CR0]
or	RA, RS, RB	OR (RS) with (RB). Place result in RA.	
or.			CR[CR0]
orc	RA, RS, RB	OR (RS) with ¬(RB). Place result in RA.	
orc.			CR[CR0]
ori	RA, RS, IM	OR (RS) with ($^{16}0$ ‖ IM). Place result in RA.	
oris	RA, RS, IM	OR (RS) with (IM ‖ $^{16}0$). Place result in RA.	

B

Instructions By Category

B.1 Instruction Set Summary – Categories

Chapter 11 (Instruction Set) contains detailed descriptions of the instructions, their operands, and notation.

Table B-1 summarizes the instruction categories in the PPC403GC instruction set. The instructions within each category are listed in subsequent tables.

Table B-1. PPC403GC Instruction Set Summary

Instruction Category	Base Instructions
Storage Reference	load, store
Arithmetic / Logical	add, subtract, negate, multiply, divide, and, or, xor, nand, nor, xnor, sign extension, count leading zeros, andc, orc
Comparison	compare algebraic, compare logical, compare immediate
Branch	branch, branch conditional, branch to LR, branch to CTR
CR Logical	crand, crnor, crxnor, crxor, crandc, crorc, crnand, cror, move cr field
Rotate/Shift	rotate and mask, rotate and insert, shift left, shift right
Cache Control	invalidate, touch, zero, flush, store, read
Interrupt Control	write to external interrupt enable bit, move to/from machine state register, return from interrupt, return from critical interrupt
TLB Management	invalidate, read entry, write entry, search, synchronize
Processor Management	system call, synchronize, trap, move to/from Device Control Registers, move to/from Special Purpose Registers, move to/from Condition Register, move to/from Machine State Register

B.2 Instructions Specific to PowerPC Embedded Controllers

To meet the functional requirements of processors for embedded systems and real-time applications, the PowerPC Embedded Controller family defines instructions that are not part of the PowerPC Architecture.

Table B-2 summarizes the PPC403GC instructions specific to the PowerPC Embedded Controller family.

Table B-2. Instructions Specific to PowerPC Embedded Controllers

Mnemonic	Operands	Function	Other Registers Changed	Page
dccci	RA, RB	Invalidate the data cache congruence class associated with the effective address (RA|0) + (RB).		11-58
dcread	RT, RA, RB	Read either tag or data information from the data cache congruence class associated with the effective address (RA|0) + (RB). Place the results in RT.		11-60
icbt	RA, RB	Load the instruction cache block which contains the effective address (RA|0) + (RB).		11-70
iccci	RA, RB	Invalidate instruction cache congruence class associated with the effective address (RA|0) + (RB).		11-72
icread	RA, RB	Read either tag or data information from the instruction cache congruence class associated with the effective address (RA|0) + (RB). Place the results in ICDBDR.		11-74
mfdcr	RT, DCRN	Move from DCR to RT, (RT) ← (DCR(DCRN)).		11-104
mtdcr	DCRN, RS	Move to DCR from RS, (DCR(DCRN)) ← (RS).		11-111
rfci		Return from critical interrupt (PC) ← (SRR2). (MSR) ← (SRR3).		11-127

Table B-2. Instructions Specific to PowerPC Embedded Controllers (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
tlbre	RT, RA,WS	If WS = 0: Load TLBHI portion of the selected TLB entry into RT. Load the PID register with the contents of the TID field of the selected TLB entry. (RT) ← TLBHI[(RA)] (PID) ← $TLB[(RA)]_{TID}$ If WS = 1: Load TLBLO portion of the selected TLB entry into RT. (RT) ← TLBLO[(RA)]		11-168
tlbsx	RT,RA,RB	Search the TLB array for a valid entry which translates the effective address EA = (RAI0) + (RB). If found, (RT) ← Index of TLB entry. If not found, (RT) Undefined.		11-170
tlbsx.		If found, (RT) ← Index of TLB entry. $CR[CR0]_{EQ}$ ← 1. If not found, (RT) Undefined. $CR[CR0]_{EQ}$ ← 1.	$CR[CR0]_{LT,GT,SO}$	
tlbwe	RS, RA,WS	If WS = 0: Write TLBHI portion of the selected TLB entry from RS. Write the TID field of the selected TLB entry from the PID register. TLBHI[(RA)] ← (RS) $TLB[(RA)]_{TID}$ ← $(PID)_{24:31}$ If WS = 1: Write TLBLO portion of the selected TLB entry from RS. TLBLO[(RA)] ← (RS)		11-172
wrtee	RS	Write value of RS_{16} to the External Enable bit (MSR[EE]).		11-180
wrteei	E	Write value of E to the External Enable bit (MSR[EE]).		11-181

B.3 Privileged Instructions

The following instructions are under control of the MSR[PR] bit, and are not allowed to be executed when MSR[PR] = b'1':

Table B-3. Privileged Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
dcbi	RA, RB	Invalidate the data cache block which contains the effective address (RAI0) + (RB).		11-50
dccci	RA, RB	Invalidate the data cache congruence class associated with the effective address (RAI0) + (RB).		11-58
dcread	RT, RA, RB	Read either tag or data information from the data cache congruence class associated with the effective address (RAI0) + (RB). Place the results in RT.		11-60
icbt	RA, RB	Load the instruction cache block which contains the effective address (RAI0) + (RB).		11-70
iccci	RA, RB	Invalidate instruction cache congruence class associated with the effective address (RAI0) + (RB).		11-72
icread	RA, RB	Read either tag or data information from the instruction cache congruence class associated with the effective address (RAI0) + (RB). Place the results in ICDBDR.		11-74
mfdcr	RT, DCRN	Move from DCR to RT, (RT) ← (DCR(DCRN)).		11-104
mfmsr	RT	Move from MSR to RT, (RT) ← (MSR).		11-106
mfspr	RT, SPRN	Move from SPR to RT, (RT) ← (SPR(SPRN)). Privileged for all SPRs except TBHU, TBLU, LR, CTR, and XER.		11-107
mtdcr	DCRN, RS	Move to DCR from RS, (DCR(DCRN)) ← (RS).		11-111
mtmsr	RS	Move to MSR from RS, (MSR) ← (RS).		11-113

Table B-3. Privileged Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
mtspr	SPRN, RS	Move to SPR from RS, (SPR(SPRN)) ← (RS). Privileged for all SPRs except LR, CTR, and XER.		11-114
rfci		Return from critical interrupt (PC) ← (SRR2). (MSR) ← (SRR3).		11-127
rfi		Return from interrupt. (PC) ← (SRR0). (MSR) ← (SRR1).		11-128
tlbia		All of the entries in the TLB are invalidated and become unavailable for translation by clearing the valid (V) bit in the TLBHI portion of each TLB entry. The rest of the fields in the TLB entries are unmodified.		11-167
tlbre	RT, RA,WS	If WS = 0: Load TLBHI portion of the selected TLB entry into RT. Load the PID register with the contents of the TID field of the selected TLB entry. (RT) ← TLBHI[(RA)] (PID) ← $TLB[(RA)]_{TID}$ If WS = 1: Load TLBLO portion of the selected TLB entry into RT. (RT) ← TLBLO[(RA)]		11-168
tlbsx	RT,RA,RB	Search the TLB array for a valid entry which translates the effective address EA = (RAI0) + (RB). If found, (RT) ← Index of TLB entry. If not found, (RT) Undefined.		11-170
tlbsx.		If found, (RT) ← Index of TLB entry. $CR[CR0]_{EQ}$ ← 1. If not found, (RT) Undefined. $CR[CR0]_{EQ}$ ← 1.	$CR[CR0]_{LT,GT,SO}$	

B

Table B-3. Privileged Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
tlbsync		**tlbsync** does not complete until all previous TLB-update instructions executed by this processor have been received and completed by all other processors. For PPC403GC, **tlbsync** is a no-op.		11-171
tlbwe	RS, RA,WS	If WS = 0: Write TLBHI portion of the selected TLB entry from RS. Write the TID field of the selected TLB entry from the PID register. TLBHI[(RA)] ← (RS) $TLB[(RA)]_{TID}$ ← $(PID)_{24:31}$ If WS = 1: Write TLBLO portion of the selected TLB entry from RS. TLBLO[(RA)] ← (RS)		11-172
wrtee	RS	Write value of RS_{16} to the External Enable bit (MSR[EE]).		11-180
wrteei	E	Write value of E to the External Enable bit (MSR[EE]).		11-181

B

B.4 Assembler Extended Mnemonics

In the appendix "Assembler Extended Mnemonics" of the PowerPC Architecture, it is required that a PowerPC assembler support at least a minimal set of extended mnemonics. These mnemonics encode to the opcodes of other instructions; the only benefit of extended mnemonics is improved usability. Code using extended mnemonics can be easier to write and to understand. Table B-4 lists the extended mnemonics required for the PPC403GC.

Note the following for every Branch Conditional mnemonic:

Bit 4 of the BO field provides a hint about the most likely outcome of a conditional branch (see Section 2.7.5 for a full discussion of Branch Prediction). Assemblers should set $BO_4 = 0$ unless a specific reason exists otherwise. In the BO field values specified in the table below, $BO_4 = 0$ has always been assumed. The assembler must allow the programmer to specify Branch Prediction. To do this, the assembler will support a suffix to every conditional branch mnemonic, as follows:

- \+ Predict branch to be taken.
- – Predict branch not to be taken.

As specific examples, **bc** also could be coded as **bc+** or **bc–**, and **bne** also could be coded **bne+** or **bne–**. These alternate codings set $BO_4 = 1$ only if the requested prediction differs from the Standard Prediction (see Section 2.7.5).

Table B-4. Extended Mnemonics for PPC403GC

Mnemonic	Operands	Function	Other Registers Changed	Page
bctr		Branch unconditionally, to address in CTR. *Extended mnemonic* for **bcctr 20,0**		11-27
bctrl		*Extended mnemonic* for **bcctrl 20,0**	(LR) ← CIA + 4.	
bdnz	target	Decrement CTR. Branch if CTR ≠ 0. *Extended mnemonic* for **bc 16,0,target**		11-20
bdnza		*Extended mnemonic* for **bca 16,0,target**		
bdnzl		*Extended mnemonic* for **bcl 16,0,target**	(LR) ← CIA + 4.	
bdnzla		*Extended mnemonic* for **bcla 16,0,target**	(LR) ← CIA + 4.	

B

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
bdnzlr		Decrement CTR. Branch if CTR ≠ 0, to address in LR. *Extended mnemonic* for **bclr 16,0**		11-31
bdnzlrl		*Extended mnemonic* for **bclrl 16,0**	(LR) ← CIA + 4.	
bdnzf	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 0. *Extended mnemonic* for **bc 0,cr_bit,target**		11-20
bdnzfa		*Extended mnemonic* for **bca 0,cr_bit,target**		
bdnzfl		*Extended mnemonic* for **bcl 0,cr_bit,target**	(LR) ← CIA + 4.	
bdnzfla		*Extended mnemonic* for **bcla 0,cr_bit,target**	(LR) ← CIA + 4.	
bdnzflr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 0, to address in LR. *Extended mnemonic* for **bclr 0,cr_bit**		11-31
bdnzflrl		*Extended mnemonic* for **bclrl 0,cr_bit**	(LR) ← CIA + 4.	
bdnzt	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 1. *Extended mnemonic* for **bc 8,cr_bit,target**		11-20
bdnzta		*Extended mnemonic* for **bca 8,cr_bit,target**		
bdnztl		*Extended mnemonic* for **bcl 8,cr_bit,target**	(LR) ← CIA + 4.	
bdnztla		*Extended mnemonic* for **bcla 8,cr_bit,target**	(LR) ← CIA + 4.	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
bdnztlr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 1, to address in LR. *Extended mnemonic* for **bclr 8,cr_bit**		11-31
bdnztlrl		*Extended mnemonic* for **bclrl 8,cr_bit**	(LR) ← CIA + 4.	
bdz	target	Decrement CTR. Branch if CTR = 0. *Extended mnemonic* for **bc 18,0,target**		11-20
bdza		*Extended mnemonic* for **bca 18,0,target**		
bdzl		*Extended mnemonic* for **bcl 18,0,target**	(LR) ← CIA + 4.	
bdzla		*Extended mnemonic* for **bcla 18,0,target**	(LR) ← CIA + 4.	
bdzlr		Decrement CTR. Branch if CTR = 0, to address in LR. *Extended mnemonic* for **bclr 18,0**		11-31
bdzlrl		*Extended mnemonic* for **bclrl 18,0**	(LR) ← CIA + 4.	
bdzf	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 0. *Extended mnemonic* for **bc 2,cr_bit,target**		11-20
bdzfa		*Extended mnemonic* for **bca 2,cr_bit,target**		
bdzfl		*Extended mnemonic* for **bcl 2,cr_bit,target**	(LR) ← CIA + 4.	
bdzfla		*Extended mnemonic* for **bcla 2,cr_bit,target**	(LR) ← CIA + 4.	

B

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
bdzflr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 0 to address in LR. *Extended mnemonic* for **bclr 2,cr_bit**		11-31
bdzflrl		*Extended mnemonic* for **bclrl 2,cr_bit**	(LR) ← CIA + 4.	
bdzt	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1. *Extended mnemonic* for **bc 10,cr_bit,target**		11-20
bdzta		*Extended mnemonic* for **bca 10,cr_bit,target**		
bdztl		*Extended mnemonic* for **bcl 10,cr_bit,target**	(LR) ← CIA + 4.	
bdztla		*Extended mnemonic* for **bcla 10,cr_bit,target**	(LR) ← CIA + 4.	
bdztlr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1, to address in LR. *Extended mnemonic* for **bclr 10,cr_bit**		11-31
bdztlrl		*Extended mnemonic* for **bclrl 10,cr_bit**	(LR) ← CIA + 4.	
beq	[cr_field,] target	Branch if equal. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bc 12,4*cr_field+2,target**		11-20
beqa		*Extended mnemonic* for **bca 12,4*cr_field+2,target**		
beql		*Extended mnemonic* for **bcl 12,4*cr_field+2,target**	(LR) ← CIA + 4.	
beqla		*Extended mnemonic* for **bcla 12,4*cr_field+2,target**	(LR) ← CIA + 4.	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
beqctr	[cr_field]	Branch if equal, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bcctr 12,4*cr_field+2**		11-27
beqctrl		*Extended mnemonic* for **bcctrl 12,4*cr_field+2**	(LR) ← CIA + 4.	
beqlr	[cr_field]	Branch if equal, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bclr 12,4*cr_field+2**		11-31
beqlrl		*Extended mnemonic* for **bclrl 12,4*cr_field+2**	(LR) ← CIA + 4.	
bf	cr_bit, target	Branch if $CR_{cr\_bit} = 0$. *Extended mnemonic* for **bc 4,cr_bit,target**		11-20
bfa		*Extended mnemonic* for **bca 4,cr_bit,target**		
bfl		*Extended mnemonic* for **bcl 4,cr_bit,target**	(LR) ← CIA + 4.	
bfla		*Extended mnemonic* for **bcla 4,cr_bit,target**	(LR) ← CIA + 4.	
bfctr	cr_bit	Branch if $CR_{cr\_bit} = 0$, to address in CTR. *Extended mnemonic* for **bcctr 4,cr_bit**		11-27
bfctrl		*Extended mnemonic* for **bcctrl 4,cr_bit**	(LR) ← CIA + 4.	
bflr	cr_bit	Branch if $CR_{cr\_bit} = 0$, to address in LR. *Extended mnemonic* for **bclr 4,cr_bit**		11-31
bflrl		*Extended mnemonic* for **bclrl 4,cr_bit**	(LR) ← CIA + 4.	

B

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
bge	[cr_field,] target	Branch if greater than or equal. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bc 4,4*cr_field+0,target**		11-20
bgea		*Extended mnemonic* for **bca 4,4*cr_field+0,target**		
bgel		*Extended mnemonic* for **bcl 4,4*cr_field+0,target**	(LR) ← CIA + 4.	
bgela		*Extended mnemonic* for **bcla 4,4*cr_field+0,target**	(LR) ← CIA + 4.	
bgectr	[cr_field]	Branch if greater than or equal, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bcctr 4,4*cr_field+0**		11-27
bgectrl		*Extended mnemonic* for **bcctrl 4,4*cr_field+0**	(LR) ← CIA + 4.	
bgelr	[cr_field]	Branch if greater than or equal, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bclr 4,4*cr_field+0**		11-31
bgelrl		*Extended mnemonic* for **bclrl 4,4*cr_field+0**	(LR) ← CIA + 4.	
bgt	[cr_field,] target	Branch if greater than. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bc 12,4*cr_field+1,target**		11-20
bgta		*Extended mnemonic* for **bca 12,4*cr_field+1,target**		
bgtl		*Extended mnemonic* for **bcl 12,4*cr_field+1,target**	(LR) ← CIA + 4.	
bgtla		*Extended mnemonic* for **bcla 12,4*cr_field+1,target**	(LR) ← CIA + 4.	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
bgtctr	[cr_field]	Branch if greater than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bcctr 12,4*cr_field+1**		11-27
bgtctrl		*Extended mnemonic* for **bcctrl 12,4*cr_field+1**	(LR) ← CIA + 4.	
bgtlr	[cr_field]	Branch if greater than, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bclr 12,4*cr_field+1**		11-31
bgtlrl		*Extended mnemonic* for **bclrl 12,4*cr_field+1**	(LR) ← CIA + 4.	
ble	[cr_field,] target	Branch if less than or equal. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bc 4,4*cr_field+1,target**		11-20
blea		*Extended mnemonic* for **bca 4,4*cr_field+1,target**		
blel		*Extended mnemonic* for **bcl 4,4*cr_field+1,target**	(LR) ← CIA + 4.	
blela		*Extended mnemonic* for **bcla 4,4*cr_field+1,target**	(LR) ← CIA + 4.	
blectr	[cr_field]	Branch if less than or equal, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bcctr 4,4*cr_field+1**		11-27
blectrl		*Extended mnemonic* for **bcctrl 4,4*cr_field+1**	(LR) ← CIA + 4.	
blelr	[cr_field]	Branch if less than or equal, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bclr 4,4*cr_field+1**		11-31
blelrl		*Extended mnemonic* for **bclrl 4,4*cr_field+1**	(LR) ← CIA + 4.	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
blr		Branch unconditionally, to address in LR. *Extended mnemonic* for **bclr 20,0**		11-31
blrl		*Extended mnemonic* for **bclrl 20,0**	(LR) ← CIA + 4.	
blt	[cr_field,] target	Branch if less than. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bc 12,4*cr_field+0,target**		11-20
blta		*Extended mnemonic* for **bca 12,4*cr_field+0,target**		
bltl		*Extended mnemonic* for **bcl 12,4*cr_field+0,target**	(LR) ← CIA + 4.	
bltla		*Extended mnemonic* for **bcla 12,4*cr_field+0,target**	(LR) ← CIA + 4.	
bltctr	[cr_field]	Branch if less than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bcctr 12,4*cr_field+0**		11-27
bltctrl		*Extended mnemonic* for **bcctrl 12,4*cr_field+0**	(LR) ← CIA + 4.	
bltlr	[cr_field]	Branch if less than, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bclr 12,4*cr_field+0**		11-31
bltlrl		*Extended mnemonic* for **bclrl 12,4*cr_field+0**	(LR) ← CIA + 4.	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
bne	[cr_field,] target	Branch if not equal. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bc 4,4*cr_field+2,target**		11-20
bnea		*Extended mnemonic* for **bca 4,4*cr_field+2,target**		
bnel		*Extended mnemonic* for **bcl 4,4*cr_field+2,target**	(LR) ← CIA + 4.	
bnela		*Extended mnemonic* for **bcla 4,4*cr_field+2,target**	(LR) ← CIA + 4.	
bnectr	[cr_field]	Branch if not equal, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bcctr 4,4*cr_field+2**		11-27
bnectrl		*Extended mnemonic* for **bcctrl 4,4*cr_field+2**	(LR) ← CIA + 4.	
bnelr	[cr_field]	Branch if not equal, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bclr 4,4*cr_field+2**		11-31
bnelrl		*Extended mnemonic* for **bclrl 4,4*cr_field+2**	(LR) ← CIA + 4.	
bng	[cr_field,] target	Branch if not greater than. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bc 4,4*cr_field+1,target**		11-20
bnga		*Extended mnemonic* for **bca 4,4*cr_field+1,target**		
bngl		*Extended mnemonic* for **bcl 4,4*cr_field+1,target**	(LR) ← CIA + 4.	
bngla		*Extended mnemonic* for **bcla 4,4*cr_field+1,target**	(LR) ← CIA + 4.	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
bngctr	[cr_field]	Branch if not greater than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bcctr 4,4*cr_field+1**		11-27
bngctrl		*Extended mnemonic* for **bcctrl 4,4*cr_field+1**	(LR) ← CIA + 4.	
bnglr	[cr_field]	Branch if not greater than, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bclr 4,4*cr_field+1**		11-31
bnglrl		*Extended mnemonic* for **bclrl 4,4*cr_field+1**	(LR) ← CIA + 4.	
bnl	[cr_field,] target	Branch if not less than. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bc 4,4*cr_field+0,target**		11-20
bnla		*Extended mnemonic* for **bca 4,4*cr_field+0,target**		
bnll		*Extended mnemonic* for **bcl 4,4*cr_field+0,target**	(LR) ← CIA + 4.	
bnlla		*Extended mnemonic* for **bcla 4,4*cr_field+0,target**	(LR) ← CIA + 4.	
bnlctr	[cr_field]	Branch if not less than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bcctr 4,4*cr_field+0**		11-27
bnlctrl		*Extended mnemonic* for **bcctrl 4,4*cr_field+0**	(LR) ← CIA + 4.	
bnllr	[cr_field]	Branch if not less than, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bclr 4,4*cr_field+0**		11-31
bnllrl		*Extended mnemonic* for **bclrl 4,4*cr_field+0**	(LR) ← CIA + 4.	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
bns	[cr_field,] target	Branch if not summary overflow. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bc 4,4*cr_field+3,target**		11-20
bnsa		*Extended mnemonic* for **bca 4,4*cr_field+3,target**		
bnsl		*Extended mnemonic* for **bcl 4,4*cr_field+3,target**	(LR) ← CIA + 4.	
bnsla		*Extended mnemonic* for **bcla 4,4*cr_field+3,target**	(LR) ← CIA + 4.	
bnsctr	[cr_field]	Branch if not summary overflow, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bcctr 4,4*cr_field+3**		11-27
bnsctrl		*Extended mnemonic* for **bcctrl 4,4*cr_field+3**	(LR) ← CIA + 4.	
bnslr	[cr_field]	Branch if not summary overflow, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bclr 4,4*cr_field+3**		11-31
bnslrl		*Extended mnemonic* for **bclrl 4,4*cr_field+3**	(LR) ← CIA + 4.	
bnu	[cr_field,] target	Branch if not unordered. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bc 4,4*cr_field+3,target**		11-20
bnua		*Extended mnemonic* for **bca 4,4*cr_field+3,target**		
bnul		*Extended mnemonic* for **bcl 4,4*cr_field+3,target**	(LR) ← CIA + 4.	
bnula		*Extended mnemonic* for **bcla 4,4*cr_field+3,target**	(LR) ← CIA + 4.	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
bnuctr	[cr_field]	Branch if not unordered, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bcctr 4,4*cr_field+3**		11-27
bnuctrl		*Extended mnemonic* for **bcctrl 4,4*cr_field+3**	(LR) ← CIA + 4.	
bnulr	[cr_field]	Branch if not unordered, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bclr 4,4*cr_field+3**		11-31
bnulrl		*Extended mnemonic* for **bclrl 4,4*cr_field+3**	(LR) ← CIA + 4.	
bso	[cr_field,] target	Branch if summary overflow. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bc 12,4*cr_field+3,target**		11-20
bsoa		*Extended mnemonic* for **bca 12,4*cr_field+3,target**		
bsol		*Extended mnemonic* for **bcl 12,4*cr_field+3,target**	(LR) ← CIA + 4.	
bsola		*Extended mnemonic* for **bcla 12,4*cr_field+3,target**	(LR) ← CIA + 4.	
bsoctr	[cr_field]	Branch if summary overflow, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bcctr 12,4*cr_field+3**		11-27
bsoctrl		*Extended mnemonic* for **bcctrl 12,4*cr_field+3**	(LR) ← CIA + 4.	
bsolr	[cr_field]	Branch if summary overflow, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bclr 12,4*cr_field+3**		11-31
bsolrl		*Extended mnemonic* for **bclrl 12,4*cr_field+3**	(LR) ← CIA + 4.	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
bt	cr_bit, target	Branch if CR_{cr_bit} = 1. *Extended mnemonic* for **bc 12,cr_bit,target**		11-20
bta		*Extended mnemonic* for **bca 12,cr_bit,target**		
btl		*Extended mnemonic* for **bcl 12,cr_bit,target**	(LR) ← CIA + 4.	
btla		*Extended mnemonic* for **bcla 12,cr_bit,target**	(LR) ← CIA + 4.	
btctr	cr_bit	Branch if CR_{cr_bit} = 1, to address in CTR. *Extended mnemonic* for **bcctr 12,cr_bit**		11-27
btctrl		*Extended mnemonic* for **bcctrl 12,cr_bit**	(LR) ← CIA + 4.	
btlr	cr_bit	Branch if CR_{cr_bit} = 1, to address in LR. *Extended mnemonic* for **bclr 12,cr_bit**		11-31
btlrl		*Extended mnemonic* for **bclrl 12,cr_bit**	(LR) ← CIA + 4.	
bun	[cr_field,] target	Branch if unordered. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bc 12,4*cr_field+3,target**		11-20
buna		*Extended mnemonic* for **bca 12,4*cr_field+3,target**		
bunl		*Extended mnemonic* for **bcl 12,4*cr_field+3,target**	(LR) ← CIA + 4.	
bunla		*Extended mnemonic* for **bcla 12,4*cr_field+3,target**	(LR) ← CIA + 4.	
bunctr	[cr_field]	Branch if unordered, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bcctr 12,4*cr_field+3**		11-27
bunctrl		*Extended mnemonic* for **bcctrl 12,4*cr_field+3**	(LR) ← CIA + 4.	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
bunlr	[cr_field]	Branch if unordered, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic* for **bclr 12,4*cr_field+3**		11-31
bunlrl		*Extended mnemonic* for **bclrl 12,4*cr_field+3**	(LR) ← CIA + 4.	
clrlwi	RA, RS, n	Clear left immediate. (n < 32) $(RA)_{0:n-1} \leftarrow {}^{n}0$ *Extended mnemonic* for **rlwinm RA,RS,0,n,31**		11-130
clrlwi.		*Extended mnemonic* for **rlwinm. RA,RS,0,n,31**	CR[CR0]	
clrslwi	RA, RS, b, n	Clear left and shift left immediate. ($n \leq b < 32$) $(RA)_{b-n:31-n} \leftarrow (RS)_{b:31}$ $(RA)_{32-n:31} \leftarrow {}^{n}0$ $(RA)_{0:b-n-1} \leftarrow {}^{b-n}0$ *Extended mnemonic* for **rlwinm RA,RS,n,b–n,31–n**		11-130
clrslwi.		*Extended mnemonic* for **rlwinm. RA,RS,n,b–n,31–n**	CR[CR0]	
clrrwi	RA, RS, n	Clear right immediate. (n < 32) $(RA)_{32-n:31} \leftarrow {}^{n}0$ *Extended mnemonic* for **rlwinm RA,RS,0,0,31–n**		11-130
clrrwi.		*Extended mnemonic* for **rlwinm. RA,RS,0,0,31–n**	CR[CR0]	
cmplw	[BF,] RA, RB	Compare Logical Word. Use CR0 if BF is omitted. *Extended mnemonic* for **cmpl BF,0,RA,RB**		11-38
cmplwi	[BF,] RA, IM	Compare Logical Word Immediate. Use CR0 if BF is omitted. *Extended mnemonic* for **cmpli BF,0,RA,IM**		11-39
cmpw	[BF,] RA, RB	Compare Word. Use CR0 if BF is omitted. *Extended mnemonic* for **cmp BF,0,RA,RB**		11-36

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
cmpwi	[BF,] RA, IM	Compare Word Immediate. Use CR0 if BF is omitted. *Extended mnemonic* for **cmpi BF,0,RA,IM**		11-37
crclr	bx	Condition register clear. *Extended mnemonic* for **crxor bx,bx,bx**		11-48
crmove	bx, by	Condition register move. *Extended mnemonic* for **cror bx,by,by**		11-46
crnot	bx, by	Condition register not. *Extended mnemonic* for **crnor bx,by,by**		11-45
crset	bx	Condition register set. *Extended mnemonic* for **creqv bx,bx,bx**		11-43
extlwi	RA, RS, n, b	Extract and left justify immediate. (n > 0) $(RA)_{0:n-1} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{n:31} \leftarrow {}^{32-n}0$ *Extended mnemonic* for **rlwinm RA,RS,b,0,n–1**		11-130
extlwi.		*Extended mnemonic* for **rlwinm. RA,RS,b,0,n–1**	CR[CR0]	
extrwi	RA, RS, n, b	Extract and right justify immediate. (n > 0) $(RA)_{32-n:31} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{0:31-n} \leftarrow {}^{32-n}0$ *Extended mnemonic* for **rlwinm RA,RS,b+n,32–n,31**		11-130
extrwi.		*Extended mnemonic* for **rlwinm. RA,RS,b+n,32–n,31**	CR[CR0]	
inslwi	RA, RS, n, b	Insert from left immediate. (n > 0) $(RA)_{b:b+n-1} \leftarrow (RS)_{0:n-1}$ *Extended mnemonic* for **rlwimi RA,RS,32–b,b,b+n–1**		11-129
inslwi.		*Extended mnemonic* for **rlwimi. RA,RS,32–b,b,b+n–1**	CR[CR0]	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
insrwi	RA, RS, n, b	Insert from right immediate. (n > 0) $(RA)_{b:b+n-1} \leftarrow (RS)_{32-n:31}$ *Extended mnemonic* for **rlwimi RA,RS,32–b–n,b,b+n–1**		11-129
insrwi.		*Extended mnemonic* for **rlwimi. RA,RS,32–b–n,b,b+n–1**	CR[CR0]	
la	RT, D(RA)	Load address. (RA ≠ 0) D is an offset from a base address that is assumed to be (RA). (RT) ← (RA) + EXTS(D) *Extended mnemonic* for **addi RT,RA,D**		11-9
li	RT, IM	Load immediate. (RT) ← EXTS(IM) *Extended mnemonic* for **addi RT,0,value**		11-9
lis	RT, IM	Load immediate shifted. (RT) ← (IM ‖ $^{16}0$) *Extended mnemonic* for **addis RT,0,value**		11-12

B

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
mfbear **mfbesr** **mfbr0** **mfbr1** **mfbr2** **mfbr3** **mfbr4** **mfbr5** **mfbr6** **mfbr7** **mfdmacc0** **mfdmacc1** **mfdmacc2** **mfdmacc3** **mfdmacr0** **mfdmacr1** **mfdmacr2** **mfdmacr3** **mfdmact0** **mfdmact1** **mfdmact2** **mfdmact3** **mfdmada0** **mfdmada1** **mfdmada2** **mfdmada3** **mfdmasa0** **mfdmasa1** **mfdmasa2** **mfdmasa3** **mfdmasr** **mfexisr** **mfexier** **mfiocr**	RT	Move from device control register DCRN. *Extended mnemonic* for **mfdcr RT,DCRN** See Table 12-3 on page 4 for listing of valid DCRN values.		11-104

B

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
mfcdbcr **mfctr** **mfdac1** **mfdac2** **mfdbsr** **mfdccr** **mfdcwr** **mfdear** **mfesr** **mfevpr** **mfiac1** **mfiac2** **mficcr** **mficdbdr** **mflr** **mfpbl1** **mfpbl2** **mfpbu1** **mfpbu2** **mfpid** **mfpit** **mfpvr** **mfsgr** **mfsprg0** **mfsprg1** **mfsprg2** **mfsprg3** **mfsrr0** **mfsrr1** **mfsrr2** **mfsrr3** **mftbhi** **mftbhu** **mftblo** **mftblu** **mftcr** **mftsr** **mfxer** **mfzpr**	RT	Move from special purpose register SPRN. *Extended mnemonic* for **mfspr RT,SPRN** See Table 12-2 on page 2 for listing of valid SPRN values.		11-107
mr	RT, RS	Move register. (RT) ← (RS) *Extended mnemonic* for **or RT,RS,RS**		11-123
mr.		*Extended mnemonic* for **or. RT,RS,RS**	CR[CR0]	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
mtcr	RS	Move to Condition Register. *Extended mnemonic* for **mtcrf 0xFF,RS**		11-109
mtbear **mtbesr** **mtbr0** **mtbr1** **mtbr2** **mtbr3** **mtbr4** **mtbr5** **mtbr6** **mtbr7** **mtdmacc0** **mtdmacc1** **mtdmacc2** **mtdmacc3** **mtdmacr0** **mtdmacr1** **mtdmacr2** **mtdmacr3** **mtdmact0** **mtdmact1** **mtdmact2** **mtdmact3** **mtdmada0** **mtdmada1** **mtdmada2** **mtdmada3** **mtdmasa0** **mtdmasa1** **mtdmasa2** **mtdmasa3** **mtdmasr** **mtexisr** **mtexier** **mtiocr**	RS	Move to device control register DCRN. *Extended mnemonic* for **mtdcr DCRN,RS** See Table 12-3 on page 4 for listing of valid DCRN values.		11-111

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
mtcdbcr **mtctr** **mtdac1** **mtdac2** **mtdbsr** **mtdccr** **mtdcwr** **mtesr** **mtevpr** **mtiac1** **mtiac2** **mticcr** **mticdbdr** **mtlr** **mtpbl1** **mtpbl2** **mtpbu1** **mtpbu2** **mtpid** **mtpit** **mtsgr** **mtsprg0** **mtsprg1** **mtsprg2** **mtsprg3** **mtsrr0** **mtsrr1** **mtsrr2** **mtsrr3** **mttbhi** **mttblo** **mttcr** **mttsr** **mtxer** **mtzpr**	RS	Move to special purpose register SPRN. *Extended mnemonic* for **mtspr SPRN,RS** See Table 12-2 on page 2 for listing of valid SPRN values.		11-114
nop		Preferred no-op, triggers optimizations based on no-ops. *Extended mnemonic* for **ori 0,0,0**		11-125

B

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
not	RA, RS	Complement register. (RA) ← ¬(RS) *Extended mnemonic* for **nor RA,RS,RS**		11-122
not.		*Extended mnemonic* for **nor. RA,RS,RS**	CR[CR0]	
rotlw	RA, RS, RB	Rotate left. (RA) ← ROTL((RS), $(RB)_{27:31}$) *Extended mnemonic* for **rlwnm RA,RS,RB,0,31**		11-133
rotlw.		*Extended mnemonic* for **rlwnm. RA,RS,RB,0,31**	CR[CR0]	
rotlwi	RA, RS, n	Rotate left immediate. (RA) ← ROTL((RS), n) *Extended mnemonic* for **rlwinm RA,RS,n,0,31**		11-130
rotlwi.		*Extended mnemonic* for **rlwinm. RA,RS,n,0,31**	CR[CR0]	
rotrwi	RA, RS, n	Rotate right immediate. (RA) ← ROTL((RS), 32–n) *Extended mnemonic* for **rlwinm RA,RS,32–n,0,31**		11-130
rotrwi.		*Extended mnemonic* for **rlwinm. RA,RS,32–n,0,31**	CR[CR0]	
slwi	RA, RS, n	Shift left immediate. (n < 32) $(RA)_{0:31-n}$ ← $(RS)_{n:31}$ $(RA)_{32-n:31}$ ← $^{n}0$ *Extended mnemonic* for **rlwinm RA,RS,n,0,31–n**		11-130
slwi.		*Extended mnemonic* for **rlwinm. RA,RS,n,0,31–n**	CR[CR0]	
srwi	RA, RS, n	Shift right immediate. (n < 32) $(RA)_{n:31}$ ← $(RS)_{0:31-n}$ $(RA)_{0:n-1}$ ← $^{n}0$ *Extended mnemonic* for **rlwinm RA,RS,32–n,n,31**		11-130
srwi.		*Extended mnemonic* for **rlwinm. RA,RS,32–n,n,31**	CR[CR0]	

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
sub	RT, RA, RB	Subtract (RB) from (RA). (RT) ← ¬(RB) + (RA) + 1. *Extended mnemonic* for **subf RT,RB,RA**		11-159
sub.		*Extended mnemonic* for **subf. RT,RB,RA**	CR[CR0]	
subo		*Extended mnemonic* for **subfo RT,RB,RA**	XER[SO, OV]	
subo.		*Extended mnemonic* for **subfo. RT,RB,RA**	CR[CR0] XER[SO, OV]	
subc	RT, RA, RB	Subtract (RB) from (RA). (RT) ← ¬(RB) + (RA) + 1. Place carry-out in XER[CA]. *Extended mnemonic* for **subfc RT,RB,RA**		11-160
subc.		*Extended mnemonic* for **subfc. RT,RB,RA**	CR[CR0]	
subco		*Extended mnemonic* for **subfco RT,RB,RA**	XER[SO, OV]	
subco.		*Extended mnemonic* for **subfco. RT,RB,RA**	CR[CR0] XER[SO, OV]	
subi	RT, RA, IM	Subtract EXTS(IM) from (RA|0). Place result in RT. *Extended mnemonic* for **addi RT,RA,–IM**		11-9
subic	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT. Place carry-out in XER[CA]. *Extended mnemonic* for **addic RT,RA,–IM**		11-10
subic.	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT. Place carry-out in XER[CA]. *Extended mnemonic* for **addic. RT,RA,–IM**	CR[CR0]	11-11
subis	RT, RA, IM	Subtract (IM || $^{16}0$) from (RA|0). Place result in RT. *Extended mnemonic* for **addis RT,RA,–IM**		11-12

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
tlbrehi	RT, RA	Load TLBHI portion of the selected TLB entry into RT. Load the PID register with the contents of the TID field of the selected TLB entry. (RT) ← TLBHI[(RA)] (PID) ← TLB[(RA)]$_{TID}$ *Extended mnemonic* for **tlbre RT,RA,0**		11-168
tlbrelo	RT, RA	Load TLBLO portion of the selected TLB entry into RT. (RT) ← TLBLO[(RA)] *Extended mnemonic* for **tlbre RT,RA,1**		11-168
tlbwehi	RS, RA	Write TLBHI portion of the selected TLB entry from RS. Write the TID field of the selected TLB entry from the PID register. TLBHI[(RA)] ← (RS) TLB[(RA)]$_{TID}$ ← (PID)$_{24:31}$ *Extended mnemonic* for **tlbwe RS,RA,0**		11-172
tlbwelo	RS, RA	Write TLBLO portion of the selected TLB entry from RS. TLBLO[(RA)] ← (RS) *Extended mnemonic* for **tlbwe RS,RA,1**		11-172

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
trap		Trap unconditionally. *Extended mnemonic* for **tw 31,0,0**		11-174
tweq	RA, RB	Trap if (RA) equal to (RB). *Extended mnemonic* for **tw 4,RA,RB**		
twge		Trap if (RA) greater than or equal to (RB). *Extended mnemonic* for **tw 12,RA,RB**		
twgt		Trap if (RA) greater than (RB). *Extended mnemonic* for **tw 8,RA,RB**		
twle		Trap if (RA) less than or equal to (RB). *Extended mnemonic* for **tw 20,RA,RB**		
twlge		Trap if (RA) logically greater than or equal to (RB). *Extended mnemonic* for **tw 5,RA,RB**		
twlgt		Trap if (RA) logically greater than (RB). *Extended mnemonic* for **tw 1,RA,RB**		
twlle		Trap if (RA) logically less than or equal to (RB). *Extended mnemonic* for **tw 6,RA,RB**		
twllt		Trap if (RA) logically less than (RB). *Extended mnemonic* for **tw 2,RA,RB**		
twlng		Trap if (RA) logically not greater than (RB). *Extended mnemonic* for **tw 6,RA,RB**		
twlnl		Trap if (RA) logically not less than (RB). *Extended mnemonic* for **tw 5,RA,RB**		
twlt		Trap if (RA) less than (RB). *Extended mnemonic* for **tw 16,RA,RB**		
twne		Trap if (RA) not equal to (RB). *Extended mnemonic* for **tw 24,RA,RB**		
twng		Trap if (RA) not greater than (RB). *Extended mnemonic* for **tw 20,RA,RB**		
twnl		Trap if (RA) not less than (RB). *Extended mnemonic* for **tw 12,RA,RB**		

Table B-4. Extended Mnemonics for PPC403GC (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
tweqi	RA, IM	Trap if (RA) equal to EXTS(IM). *Extended mnemonic* for **twi 4,RA,IM**		11-177
twgei		Trap if (RA) greater than or equal to EXTS(IM). *Extended mnemonic* for **twi 12,RA,IM**		
twgti		Trap if (RA) greater than EXTS(IM). *Extended mnemonic* for **twi 8,RA,IM**		
twlei		Trap if (RA) less than or equal to EXTS(IM). *Extended mnemonic* for **twi 20,RA,IM**		
twlgei		Trap if (RA) logically greater than or equal to EXTS(IM). *Extended mnemonic* for **twi 5,RA,IM**		
twlgti		Trap if (RA) logically greater than EXTS(IM). *Extended mnemonic* for **twi 1,RA,IM**		
twllei		Trap if (RA) logically less than or equal to EXTS(IM). *Extended mnemonic* for **twi 6,RA,IM**		
twllti		Trap if (RA) logically less than EXTS(IM). *Extended mnemonic* for **twi 2,RA,IM**		
twlngi		Trap if (RA) logically not greater than EXTS(IM). *Extended mnemonic* for **twi 6,RA,IM**		
twlnli		Trap if (RA) logically not less than EXTS(IM). *Extended mnemonic* for **twi 5,RA,IM**		
twlti		Trap if (RA) less than EXTS(IM). *Extended mnemonic* for **twi 16,RA,IM**		
twnei		Trap if (RA) not equal to EXTS(IM). *Extended mnemonic* for **twi 24,RA,IM**		
twngi		Trap if (RA) not greater than EXTS(IM). *Extended mnemonic* for **twi 20,RA,IM**		
twnli		Trap if (RA) not less than EXTS(IM). *Extended mnemonic* for **twi 12,RA,IM**		

B.5 Storage Reference Instructions

The PPC403GC uses load and store instructions to transfer data between memory and the general purpose registers. Load and store instructions operate on byte, halfword and word data. The Storage Reference instructions also support loading or storing multiple registers, character strings, and byte-reversed data. Table B-5 shows the Storage Reference instructions available for use in the PPC403GC.

Table B-5. Storage Reference Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
lbz	RT, D(RA)	Load byte from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1).		11-77
lbzu	RT, D(RA)	Load byte from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1). Update the base address, (RA) ← EA.		11-78
lbzux	RT, RA, RB	Load byte from EA = (RAI0) + (RB) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1). Update the base address, (RA) ← EA.		11-79
lbzx	RT, RA, RB	Load byte from EA = (RAI0) + (RB) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1).		11-80
lha	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and sign extend, (RT) ← EXTS(MS(EA,2)).		11-81
lhau	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and sign extend, (RT) ← EXTS(MS(EA,2)). Update the base address, (RA) ← EA.		11-82
lhaux	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) and sign extend, (RT) ← EXTS(MS(EA,2)). Update the base address, (RA) ← EA.		11-83
lhax	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) and sign extend, (RT) ← EXTS(MS(EA,2)).		11-84

Table B-5. Storage Reference Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
lhbrx	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) then reverse byte order and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA+1,1) || MS(EA,1).		11-85
lhz	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2).		11-86
lhzu	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2). Update the base address, (RA) ← EA.		11-87
lhzux	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2). Update the base address, (RA) ← EA.		11-88
lhzx	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2).		11-89
lmw	RT, D(RA)	Load multiple words starting from EA = (RAI0) + EXTS(D). Place into consecutive registers, RT through GPR(31). RA is not altered unless RA = GPR(31).		11-90
lswi	RT, RA, NB	Load consecutive bytes from EA=(RAI0). Number of bytes n=32 if NB=0, else n=NB. Stack bytes into words in CEIL(n/4) consecutive registers starting with RT, to R_{FINAL} ← ((RT + CEIL(n/4) – 1) % 32). GPR(0) is consecutive to GPR(31). RA is not altered unless RA = R_{FINAL}.		11-91
lswx	RT, RA, RB	Load consecutive bytes from EA=(RAI0)+(RB). Number of bytes n=XER[TBC]. Stack bytes into words in CEIL(n/4) consecutive registers starting with RT, to R_{FINAL} ← ((RT + CEIL(n/4) – 1) % 32). GPR(0) is consecutive to GPR(31). RA is not altered unless RA = R_{FINAL}. RB is not altered unless RB = R_{FINAL}. If n=0, content of RT is undefined.		11-93

Table B-5. Storage Reference Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
lwarx	RT, RA, RB	Load word from EA = (RAI0) + (RB) and place in RT, (RT) ← MS(EA,4). Set the Reservation bit.		11-95
lwbrx	RT, RA, RB	Load word from EA = (RAI0) + (RB) then reverse byte order, (RT) ← MS(EA+3,1) || MS(EA+2,1) || MS(EA+1,1) || MS(EA,1).		11-96
lwz	RT, D(RA)	Load word from EA = (RAI0) + EXTS(D) and place in RT, (RT) ← MS(EA,4).		11-97
lwzu	RT, D(RA)	Load word from EA = (RAI0) + EXTS(D) and place in RT, (RT) ← MS(EA,4). Update the base address, (RA) ← EA.		11-98
lwzux	RT, RA, RB	Load word from EA = (RAI0) + (RB) and place in RT, (RT) ← MS(EA,4). Update the base address, (RA) ← EA.		11-99
lwzx	RT, RA, RB	Load word from EA = (RAI0) + (RB) and place in RT, (RT) ← MS(EA,4).		11-100
stb	RS, D(RA)	Store byte $(RS)_{24:31}$ in memory at EA = (RAI0) + EXTS(D).		11-139
stbu	RS, D(RA)	Store byte $(RS)_{24:31}$ in memory at EA = (RAI0) + EXTS(D). Update the base address, (RA) ← EA.		11-140
stbux	RS, RA, RB	Store byte $(RS)_{24:31}$ in memory at EA = (RAI0) + (RB). Update the base address, (RA) ← EA.		11-141
stbx	RS, RA, RB	Store byte $(RS)_{24:31}$ in memory at EA = (RAI0) + (RB).		11-142
sth	RS, D(RA)	Store halfword $(RS)_{16:31}$ in memory at EA = (RAI0) + EXTS(D).		11-143

Table B-5. Storage Reference Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
sthbrx	RS, RA, RB	Store halfword $(RS)_{16:31}$ byte-reversed in memory at EA = (RAI0) + (RB). MS(EA, 2) ← $(RS)_{24:31}$ ‖ $(RS)_{16:23}$		11-144
sthu	RS, D(RA)	Store halfword $(RS)_{16:31}$ in memory at EA = (RAI0) + EXTS(D). Update the base address, (RA) ← EA.		11-145
sthux	RS, RA, RB	Store halfword $(RS)_{16:31}$ in memory at EA = (RAI0) + (RB). Update the base address, (RA) ← EA.		11-146
sthx	RS, RA, RB	Store halfword $(RS)_{16:31}$ in memory at EA = (RAI0) + (RB).		11-147
stmw	RS, D(RA)	Store consecutive words from RS through GPR(31) in memory starting at EA = (RAI0) + EXTS(D).		11-148
stswi	RS, RA, NB	Store consecutive bytes in memory starting at EA=(RAI0). Number of bytes n=32 if NB=0, else n=NB. Bytes are unstacked from CEIL(n/4) consecutive registers starting with RS. GPR(0) is consecutive to GPR(31).		11-149
stswx	RS, RA, RB	Store consecutive bytes in memory starting at EA=(RAI0)+(RB). Number of bytes n=XER[TBC]. Bytes are unstacked from CEIL(n/4) consecutive registers starting with RS. GPR(0) is consecutive to GPR(31).		11-150
stw	RS, D(RA)	Store word (RS) in memory at EA = (RAI0) + EXTS(D).		11-152
stwbrx	RS, RA, RB	Store word (RS) byte-reversed in memory at EA = (RAI0) + (RB). MS(EA, 4) ← $(RS)_{24:31}$ ‖ $(RS)_{16:23}$ ‖ $(RS)_{8:15}$ ‖ $(RS)_{0:7}$		11-153

Table B-5. Storage Reference Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page

stwcx.	RS, RA, RB	Store word (RS) in memory at
		EA = (RAI0) + (RB)
		only if reservation bit is set.
		if RESERVE = 1 then
		MS(EA, 4) ← (RS)
		RESERVE ← 0
		(CR[CR0]) ← $^{2}0$ || 1 || XER_{SO}
		else
		(CR[CR0]) ← $^{2}0$ || 0 || XER_{SO}.
				11-154

stwu	RS, D(RA)	Store word (RS) in memory at EA = (RAI0) + EXTS(D). Update the base address, (RA) ← EA.		11-156
stwux	RS, RA, RB	Store word (RS) in memory at EA = (RAI0) + (RB). Update the base address, (RA) ← EA.		11-157
stwx	RS, RA, RB	Store word (RS) in memory at EA = (RAI0) + (RB).		11-158

B.6 Arithmetic and Logical Instructions

Table B-6 shows the set of arithmetic and logical instructions supported by the PPC403GC. Arithmetic operations are performed on integer or ordinal operands stored in registers. Instructions using two operands are defined in a three operand format where the operation is performed on the operands stored in two registers and the result is placed in a third register. Instructions using one operand are defined in a two operand format where the operation is performed on the operand in one register and the result is placed in another register. Several instructions also have immediate formats in which one operand is coded as part of the instruction itself. Most arithmetic and logical instructions can optionally set the condition code register based on the outcome of the instruction.

Table B-6. Arithmetic and Logical Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
add	RT, RA, RB	Add (RA) to (RB). Place result in RT.		11-6
add.			CR[CR0]	
addo			XER[SO, OV]	
addo.			CR[CR0] XER[SO, OV]	
addc	RT, RA, RB	Add (RA) to (RB). Place result in RT. Place carry-out in XER[CA].		11-7
addc.			CR[CR0]	
addco			XER[SO, OV]	
addco.			CR[CR0] XER[SO, OV]	
adde	RT, RA, RB	Add XER[CA], (RA), (RB). Place result in RT. Place carry-out in XER[CA].		11-8
adde.			CR[CR0]	
addeo			XER[SO, OV]	
addeo.			CR[CR0] XER[SO, OV]	
addi	RT, RA, IM	Add EXTS(IM) to (RAI0). Place result in RT.		11-9
addic	RT, RA, IM	Add EXTS(IM) to (RAI0). Place result in RT. Place carry-out in XER[CA].		11-10
addic.	RT, RA, IM	Add EXTS(IM) to (RAI0). Place result in RT. Place carry-out in XER[CA].	CR[CR0]	11-11

Table B-6. Arithmetic and Logical Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
addis	RT, RA, IM	Add (IM || ^{16}0) to (RAI0). Place result in RT.		11-12
addme	RT, RA	Add XER[CA], (RA), (-1). Place result in RT. Place carry-out in XER[CA].		11-13
addme.			CR[CR0]	
addmeo			XER[SO, OV]	
addmeo.			CR[CR0] XER[SO, OV]	
addze	RT, RA	Add XER[CA] to (RA). Place result in RT. Place carry-out in XER[CA].		11-14
addze.			CR[CR0]	
addzeo			XER[SO, OV]	
addzeo.			CR[CR0] XER[SO, OV]	
and	RA, RS, RB	AND (RS) with (RB). Place result in RA.		11-15
and.			CR[CR0]	
andc	RA, RS, RB	AND (RS) with ¬(RB). Place result in RA.		11-16
andc.			CR[CR0]	
andi.	RA, RS, IM	AND (RS) with (^{16}0 || IM). Place result in RA.	CR[CR0]	11-17
andis.	RA, RS, IM	AND (RS) with (IM || ^{16}0). Place result in RA.	CR[CR0]	11-18
cntlzw	RA, RS	Count leading zeros in RS. Place result in RA.		11-40
cntlzw.			CR[CR0]	
divw	RT, RA, RB	Divide (RA) by (RB), signed. Place result in RT.		11-62
divw.			CR[CR0]	
divwo			XER[SO, OV]	
divwo.			CR[CR0] XER[SO, OV]	

B

Table B-6. Arithmetic and Logical Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
divwu	RT, RA, RB	Divide (RA) by (RB), unsigned. Place result in RT.		11-63
divwu.			CR[CR0]	
divwuo			XER[SO, OV]	
divwuo.			CR[CR0] XER[SO, OV]	
eqv	RA, RS, RB	Equivalence of (RS) with (RB). (RA) ← ¬((RS) ⊕ (RB))		11-65
eqv.			CR[CR0]	
extsb	RA, RS	Extend the sign of byte $(RS)_{24:31}$. Place the result in RA.		11-66
extsb.			CR[CR0]	
extsh	RA, RS	Extend the sign of halfword $(RS)_{16:31}$. Place the result in RA.		11-67
extsh.			CR[CR0]	
mulhw	RT, RA, RB	Multiply (RA) and (RB), signed. Place hi-order result in RT. $prod_{0:63}$ ← (RA) × (RB) (signed). (RT) ← $prod_{0:31}$.		11-116
mulhw.			CR[CR0]	
mulhwu	RT, RA, RB	Multiply (RA) and (RB), unsigned. Place hi-order result in RT. $prod_{0:63}$ ← (RA) × (RB) (unsigned). (RT) ← $prod_{0:31}$.		11-117
mulhwu.			CR[CR0]	
mulli	RT, RA, IM	Multiply (RA) and IM, signed. Place lo-order result in RT. $prod_{0:47}$ ← (RA) × IM (signed) (RT) ← $prod_{16:47}$		11-118
mullw	RT, RA, RB	Multiply (RA) and (RB), signed. Place lo-order result in RT. $prod_{0:63}$ ← (RA) × (RB) (signed). (RT) ← $prod_{32:63}$.		11-119
mullw.			CR[CR0]	
mullwo			XER[SO, OV]	
mullwo.			CR[CR0] XER[SO, OV]	
nand	RA, RS, RB	NAND (RS) with (RB). Place result in RA.		11-120
nand.			CR[CR0]	

B

Table B-6. Arithmetic and Logical Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
neg	RT, RA	Negative (two's complement) of RA. (RT) ← ¬(RA) + 1		11-121
neg.			CR[CR0]	
nego			XER[SO, OV]	
nego.			CR[CR0] XER[SO, OV]	
nor	RA, RS, RB	NOR (RS) with (RB). Place result in RA.		11-122
nor.			CR[CR0]	
or	RA, RS, RB	OR (RS) with (RB). Place result in RA.		11-123
or.			CR[CR0]	
orc	RA, RS, RB	OR (RS) with ¬(RB). Place result in RA.		11-124
orc.			CR[CR0]	
ori	RA, RS, IM	OR (RS) with ($^{16}0$ || IM). Place result in RA.		11-125
oris	RA, RS, IM	OR (RS) with (IM || $^{16}0$). Place result in RA.		11-126
subf	RT, RA, RB	Subtract (RA) from (RB). (RT) ← ¬(RA) + (RB) + 1.		11-159
subf.			CR[CR0]	
subfo			XER[SO, OV]	
subfo.			CR[CR0] XER[SO, OV]	
subfc	RT, RA, RB	Subtract (RA) from (RB). (RT) ← ¬(RA) + (RB) + 1. Place carry-out in XER[CA].		11-160
subfc.			CR[CR0]	
subfco			XER[SO, OV]	
subfco.			CR[CR0] XER[SO, OV]	
subfe	RT, RA, RB	Subtract (RA) from (RB) with carry-in. (RT) ← ¬(RA) + (RB) + XER[CA]. Place carry-out in XER[CA].		11-162
subfe.			CR[CR0]	
subfeo			XER[SO, OV]	
subfeo.			CR[CR0] XER[SO, OV]	

Table B-6. Arithmetic and Logical Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
subfic	RT, RA, IM	Subtract (RA) from EXTS(IM). (RT) ← ¬(RA) + EXTS(IM) + 1. Place carry-out in XER[CA].		11-163
subfme	RT, RA, RB	Subtract (RA) from (–1) with carry-in. (RT) ← ¬(RA) + (–1) + XER[CA]. Place carry-out in XER[CA].		11-164
subfme.			CR[CR0]	
subfmeo			XER[SO, OV]	
subfmeo.			CR[CR0] XER[SO, OV]	
subfze	RT, RA, RB	Subtract (RA) from zero with carry-in. (RT) ← ¬(RA) + XER[CA]. Place carry-out in XER[CA].		11-165
subfze.			CR[CR0]	
subfzeo			XER[SO, OV]	
subfzeo.			CR[CR0] XER[SO, OV]	
xor	RA, RS, RB	XOR (RS) with (RB). Place result in RA.		11-182
xor.			CR[CR0]	
xori	RA, RS, IM	XOR (RS) with ($^{16}0$ || IM). Place result in RA.		11-183
xoris	RA, RS, IM	XOR (RS) with (IM || $^{16}0$). Place result in RA.		11-184

B.7 Condition Register Logical Instructions

Condition Register (CR) logical instructions allow the user to combine the results of several comparisons without incurring the overhead of conditional branching. These instructions can significantly improve code performance if multiple conditions are tested prior to making a branch decision. Table B-7 summarizes the CR logical instructions.

Table B-7. Condition Register Logical Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
crand	BT, BA, BB	AND bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		11-41
crandc	BT, BA, BB	AND bit (CR_{BA}) with $\neg$(CR_{BB}). Place result in CR_{BT}.		11-42
creqv	BT, BA, BB	Equivalence of bit CR_{BA} with CR_{BB}. $CR_{BT} \leftarrow \neg(CR_{BA} \oplus CR_{BB})$		11-43
crnand	BT, BA, BB	NAND bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		11-44
crnor	BT, BA, BB	NOR bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		11-45
cror	BT, BA, BB	OR bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		11-46
crorc	BT, BA, BB	OR bit (CR_{BA}) with $\neg$(CR_{BB}). Place result in CR_{BT}.		11-47
crxor	BT, BA, BB	XOR bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		11-48
mcrf	BF, BFA	Move CR field, (CR[CRn]) $\leftarrow$ (CR[CRm]) where m $\leftarrow$ BFA and n $\leftarrow$ BF.		11-101

B.8 Branch Instructions

The architecture provides conditional and unconditional branches to any storage location. The conditional branch instructions test condition codes set previously and branch accordingly. Conditional branch instructions may decrement and test the Count Register (CTR) as part of determination of the branch condition and may save the return address in the Link Register (LR). The target address for a branch may be a displacement from the current instruction address (CIA), or may be contained in the LR or CTR, or may be an absolute address.

Table B-8. Branch Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
b	target	Branch unconditional relative. LI ← (target – CIA)$_{6:29}$ NIA ← CIA + EXTS(LI || $^{2}0$)		11-19
ba		Branch unconditional absolute. LI ← target$_{6:29}$ NIA ← EXTS(LI || $^{2}0$)		
bl		Branch unconditional relative. LI ← (target – CIA)$_{6:29}$ NIA ← CIA + EXTS(LI || $^{2}0$)	(LR) ← CIA + 4.	
bla		Branch unconditional absolute. LI ← target$_{6:29}$ NIA ← EXTS(LI || $^{2}0$)	(LR) ← CIA + 4.	
bc	BO, BI, target	Branch conditional relative. BD ← (target – CIA)$_{16:29}$ NIA ← CIA + EXTS(BD || $^{2}0$)	CTR if BO_2 = 0.	11-20
bca		Branch conditional absolute. BD ← target$_{16:29}$ NIA ← EXTS(BD || $^{2}0$)	CTR if BO_2 = 0.	
bcl		Branch conditional relative. BD ← (target – CIA)$_{16:29}$ NIA ← CIA + EXTS(BD || $^{2}0$)	CTR if BO_2 = 0. (LR) ← CIA + 4.	
bcla		Branch conditional absolute. BD ← target$_{16:29}$ NIA ← EXTS(BD || $^{2}0$)	CTR if BO_2 = 0. (LR) ← CIA + 4.	
bcctr	BO, BI	Branch conditional to address in CTR. Using (CTR) at exit from instruction, NIA ← CTR$_{0:29}$ || $^{2}0$.	CTR if BO_2 = 0.	11-27
bcctrl			CTR if BO_2 = 0. (LR) ← CIA + 4.	

Table B-8. Branch Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
bclr	BO, BI	Branch conditional to address in LR. Using (LR) at entry to instruction, NIA ← $LR_{0:29}$ || $^{2}0$.	CTR if $BO_2 = 0$.	11-31
bclrl			CTR if $BO_2 = 0$. (LR) ← CIA + 4.	

B.9 Comparison Instructions

Comparison instructions perform arithmetic and logical comparisons between two operands and set one of the eight condition code register fields based on the outcome of the comparison. Table B-9 shows the comparison instructions supported by the PPC403GC.

Table B-9. Comparison Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
cmp	BF, 0, RA, RB	Compare (RA) to (RB), signed. Results in CR[CRn], where n = BF.		11-36
cmpi	BF, 0, RA, IM	Compare (RA) to EXTS(IM), signed. Results in CR[CRn], where n = BF.		11-37
cmpl	BF, 0, RA, RB	Compare (RA) to (RB), unsigned. Results in CR[CRn], where n = BF.		11-38
cmpli	BF, 0, RA, IM	Compare (RA) to ($^{16}0$ || IM), unsigned. Results in CR[CRn], where n = BF.		11-39

B.10 Rotate and Shift Instructions

Rotate and shift instructions rotate and shift operands which are stored in the general purpose registers. Rotate instructions can also mask rotated operands. Table B-10 shows the PPC403GC rotate and shift instructions.

Table B-10. Rotate and Shift Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
rlwimi **rlwimi.**	RA, RS, SH, MB, ME	Rotate left word immediate, then insert according to mask. r ← ROTL((RS), SH) m ← MASK(MB, ME) (RA) ← (r ∧ m) ∨ ((RA) ∧ ¬m)	 CR[CR0]	11-129
rlwinm **rlwinm.**	RA, RS, SH, MB, ME	Rotate left word immediate, then AND with mask. r ← ROTL((RS), SH) m ← MASK(MB, ME) (RA) ← (r ∧ m)	 CR[CR0]	11-130
rlwnm **rlwnm.**	RA, RS, RB, MB, ME	Rotate left word, then AND with mask. r ← ROTL((RS), $(RB)_{27:31}$) m ← MASK(MB, ME) (RA) ← (r ∧ m)	 CR[CR0]	11-133
slw **slw.**	RA, RS, RB	Shift left (RS) by $(RB)_{27:31}$. n ← $(RB)_{27:31}$. r ← ROTL((RS), n). if $(RB)_{26}$ = 0 then m ← MASK(0, 31 – n) else m ← $^{32}0$. (RA) ← r ∧ m.	 CR[CR0]	11-135
sraw **sraw.**	RA, RS, RB	Shift right algebraic (RS) by $(RB)_{27:31}$. n ← $(RB)_{27:31}$. r ← ROTL((RS), 32 – n). if $(RB)_{26}$ = 0 then m ← MASK(n, 31) else m ← $^{32}0$. s ← $(RS)_0$. (RA) ← (r ∧ m) ∨ (^{32}s ∧ ¬m). XER[CA] ← s ∧ ((r ∧ ¬m) ≠ 0).	 CR[CR0]	11-136
srawi **srawi.**	RA, RS, SH	Shift right algebraic (RS) by SH. n ← SH. r ← ROTL((RS), 32 – n). m ← MASK(n, 31). s ← $(RS)_0$. (RA) ← (r ∧ m) ∨ (^{32}s ∧ ¬m). XER[CA] ← s ∧ ((r ∧ ¬m)≠0).	 CR[CR0]	11-137

B

Table B-10. Rotate and Shift Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
srw	RA, RS, RB	Shift right (RS) by $(RB)_{27:31}$. $n \leftarrow (RB)_{27:31}$. $r \leftarrow$ ROTL((RS), 32 – n). if $(RB)_{26}$ = 0 then m ← MASK(n, 31) else m ← $^{32}0$. (RA) ← r ∧ m.		11-138
srw.			CR[CR0]	

B.11 Cache Control Instructions

Cache control instructions allow the user to indirectly control the contents of the data and instruction caches. The user may fill, flush, invalidate and zero blocks (16-byte lines) in the data cache. The user may also invalidate congruence classes in both caches and invalidate individual lines in the instruction cache.

Table B-11. Cache Control Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
dcbf	RA, RB	Flush (store, then invalidate) the data cache block which contains the effective address (RAI0) + (RB).		11-49
dcbi	RA, RB	Invalidate the data cache block which contains the effective address (RAI0) + (RB).		11-50
dcbst	RA, RB	Store the data cache block which contains the effective address (RAI0) + (RB).		11-51
dcbt	RA, RB	Load the data cache block which contains the effective address (RAI0) + (RB).		11-52
dcbtst	RA,RB	Load the data cache block which contains the effective address (RAI0) + (RB).		11-54
dcbz	RA, RB	Zero the data cache block which contains the effective address (RAI0) + (RB).		11-56
dccci	RA, RB	Invalidate the data cache congruence class associated with the effective address (RAI0) + (RB).		11-58
dcread	RT, RA, RB	Read either tag or data information from the data cache congruence class associated with the effective address (RAI0) + (RB). Place the results in RT.		11-60
icbi	RA, RB	Invalidate the instruction cache block which contains the effective address (RAI0) + (RB).		11-68
icbt	RA, RB	Load the instruction cache block which contains the effective address (RAI0) + (RB).		11-70
iccci	RA, RB	Invalidate instruction cache congruence class associated with the effective address (RAI0) + (RB).		11-72

Table B-11. Cache Control Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
icread	RA, RB	Read either tag or data information from the instruction cache congruence class associated with the effective address (RAI0) + (RB). Place the results in ICDBDR.		11-74

B.12 Interrupt Control Instructions

The interrupt control instructions allow the user to move data between general purpose registers and the machine state register, return from interrupts and enable or disable maskable external interrupts. Table B-12 shows the Interrupt control instruction set.

Table B-12. Interrupt Control Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
mfmsr	RT	Move from MSR to RT, (RT) ← (MSR).		11-106
mtmsr	RS	Move to MSR from RS, (MSR) ← (RS).		11-113
rfci		Return from critical interrupt (PC) ← (SRR2). (MSR) ← (SRR3).		11-127
rfi		Return from interrupt. (PC) ← (SRR0). (MSR) ← (SRR1).		11-128
wrtee	RS	Write value of RS_{16} to the External Enable bit (MSR[EE]).		11-180
wrteei	E	Write value of E to the External Enable bit (MSR[EE]).		11-181

B.13 TLB Management Instructions

The TLB management instructions read and write entries of the TLB array in the MMU, search the TLB array for an entry which will translate a given address, invalidate all TLB entries, and synchronize TLB updates with other processors.

Table B-13. TLB Management Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
tlbia		All of the entries in the TLB are invalidated and become unavailable for translation by clearing the valid (V) bit in the TLBHI portion of each TLB entry. The rest of the fields in the TLB entries are unmodified.		11-167
tlbre	RT, RA,WS	If WS = 0: Load TLBHI portion of the selected TLB entry into RT. Load the PID register with the contents of the TID field of the selected TLB entry. (RT) ← TLBHI[(RA)] (PID) ← $TLB[(RA)]_{TID}$ If WS = 1: Load TLBLO portion of the selected TLB entry into RT. (RT) ← TLBLO[(RA)]		11-168
tlbsx	RT,RA,RB	Search the TLB array for a valid entry which translates the effective address EA = (RAI0) + (RB). If found, (RT) ← Index of TLB entry. If not found, (RT) Undefined.		11-170
tlbsx.		If found, (RT) ← Index of TLB entry. $CR[CR0]_{EQ}$ ← 1. If not found, (RT) Undefined. $CR[CR0]_{EQ}$ ← 1.	$CR[CR0]_{LT,GT,SO}$	
tlbsync		**tlbsync** does not complete until all previous TLB-update instructions executed by this processor have been received and completed by all other processors. For PPC403GC, **tlbsync** is a no-op.		11-171

Table B-13. TLB Management Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
tlbwe	RS, RA,WS	If WS = 0: Write TLBHI portion of the selected TLB entry from RS. Write the TID field of the selected TLB entry from the PID register. TLBHI[(RA)] ← (RS) $TLB[(RA)]_{TID} \leftarrow (PID)_{24:31}$ If WS = 1: Write TLBLO portion of the selected TLB entry from RS. TLBLO[(RA)] ← (RS)		11-172

B.14 Processor Management Instructions

The processor management instructions move data between GPRs and SPRs and DCRs in the PPC403GC; these instructions also provide traps, system calls and synchronization controls.

Table B-14. Processor Management Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
eieio		Storage synchronization. All loads and stores that precede the **eieio** instruction complete before any loads and stores that follow the instruction access main storage. Implemented as **sync**, which is more restrictive.		11-64
isync		Synchronize execution context by flushing the prefetch queue.		11-76
mcrxr	BF	Move XER[0:3] into field CRn, where n←BF. CR[CRn] ← (XER[SO, OV, CA]). (XER[SO, OV, CA]) ← $^{3}0$.		11-102
mfcr	RT	Move from CR to RT, (RT) ← (CR).		11-103
mfdcr	RT, DCRN	Move from DCR to RT, (RT) ← (DCR(DCRN)).		11-104
mfspr	RT, SPRN	Move from SPR to RT, (RT) ← (SPR(SPRN)).		11-107
mtcrf	FXM, RS	Move some or all of the contents of RS into CR as specified by FXM field, mask ← $^{4}(FXM_0)$ || $^{4}(FXM_1)$ || ... || $^{4}(FXM_6)$ || $^{4}(FXM_7)$. (CR)←((RS) ∧ mask) ∨ (CR) ∧ ¬mask).		11-109
mtdcr	DCRN, RS	Move to DCR from RS, (DCR(DCRN)) ← (RS).		11-111
mtspr	SPRN, RS	Move to SPR from RS, (SPR(SPRN)) ← (RS).		11-114
sc		System call exception is generated. (SRR1) ← (MSR) (SRR0) ← (PC) PC ← $EVPR_{0:15}$ || x'0C00' (MSR[WE, PR, EE, PE, DR, IR]) ← 0 (MSR[LE]) ← (MSR[ILE])		11-134

Table B-14. Processor Management Instructions (cont.)

Mnemonic	Operands	Function	Other Registers Changed	Page
sync		Synchronization. All instructions that precede **sync** complete before any instructions that follow **sync** begin. When **sync** completes, all storage accesses initiated prior to **sync** will have completed.		11-166
tw	TO, RA, RB	Trap exception is generated if, comparing (RA) with (RB), any condition specified by TO is true.		11-174
twi	TO, RA, IM	Trap exception is generated if, comparing (RA) with EXTS(IM), any condition specified by TO is true.		11-177