Open8 Assembly Language Reference Manual

Revision 159

, May 25, 2020

This Page Intentionally Left Blank.

Modified BSD License

Copyright (c) 2011, Kirk I. Hays
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- **Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.**
- **Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.**
- **Neither the name of Kirk I. Hays nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.**

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This Page Intentionally Left Blank.

Table of Contents

Introduction

The Open8 softcore, developed by Jeremy Seth Henry in 2006, is a “clean-room” derivation of the Vautomation Inc. V8-uRISC™ processor, introduced in the mid-1990s, and later sold by ARC International as the ARClite™ microRISC™.

Vautomation was acquired by ARC International in 2002. ARC International, in turn, was acquired by Virage Logic in 2009, which was then acquired by Synopsus in 2010.

The Open8 core is an 8-bit microprocessor with 16 bit addressing, eight 8-bit registers, and a 64K address space.

This document covers the instruction set as supported by the GNU *binutils* tools, as ported by Kirk I. Hays, and provides the ABI implemented by the GNU *gcc* compiler suite.

This Page Intentionally Left Blank.

Instruction Mnemonics

The instructions documented in this section are executable hardware instructions. Convenience mnemonics ("Pseudo-mnemonics") supported by the assembler are documented in the section Instruction Pseudo-Mnemonics.

Notation

The general purpose registers (*R0..R7*) are represented as *Rn*.

Even numbered registers, where required, are represented as *Re*. Generally, when an even numbered register is required, it is implied that the next higher numbered odd register is also used by the operation.

If a pair of registers is required for an operation, then the concatenation of those registers is indicated by braces, such as *{R1:R0}*. The lower order bits are in the lower numbered register.

If a particular general purpose register is necessary, it is specified directly, such as *R0*.

Optional elements within the specification of a mnemonic format are enclosed in brackets.

If a particular bit of a register is needed, it is specified with an index in brackets following the register name (for example, *Rn[7]*), where bits in a register are numbered from least significant to most significant. As a special case, the carry bit from a mathematical operation is indicated as *Rn[8]* for 8-bit math, and as *{Re+1:Re}[16]* for 16 bit math.

Assignment of a value to a register (or bit) is indicated with the "receives" operator, "←".

The arithmetic operators have their usual meanings when used on values from registers.

Mathematical operations are unsigned by default. Where signed operations or values are needed, they are called out explicitly.

C-style comparison operators, such as "==" are used, and have their standard meanings. They produce a single bit result, "1" for true, and "0" for false.

PSR is the program status register.

STACK is the stack pointer register.

PROGRAM_COUNTER is the pointer to the currently executing instruction.

VECTOR_BASE is the implementation specific location of the interrupt vectors.

INC – Increment Register

Format

Opcode					Reg		
0	0	0	0	0	X	X	X

Mnemonic

[label:] INC R*n*

Operation

Rn ← Rn + 1

PSR[2] ← Rn[7]

PSR[1] ← Rn[8]

PSR[0] ← (Rn == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	X	X

Clocks

1

Description

Increment the value in a general purpose register.

Negative, Carry, and Zero Flags are set to reflect the new value of R*n*.

ADC – Add Register and Carry Flag to Accumulator

Format

Opcode					Reg		
0	0	0	0	1	X	X	X

Mnemonic

[label:] ADC R*n*

Operation

R0 ← R0 + Rn + PSR[1]

PSR[2] ← R0[7]

PSR[1] ← R0[8]

PSR[0] ← (R0 == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	X	X

Clocks

1

Description

Arithmetically add a general purpose register and the current value of the carry flag to the accumulator. Negative, Carry, and Zero Flags are set to reflect the new value of R0.

TX0 – Transfer Register to Accumulator

Format

Opcode					Reg		
0	0	0	1	0	X	X	X

Mnemonic

[label:] TX0 R*n*

Operation

R0 ← Rn

PSR[2] ← R0[7]

PSR[0] ← (R0 == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	-	X

Clocks

1

Description

Transfer the contents of a general purpose register to the accumulator.

Negative and Zero Flags are set to reflect the new value of R0.

OR – Bitwise OR Register to Accumulator

Format

Opcode					Reg		
0	0	0	1	1	X	X	X

Mnemonic

[label:] OR R*n*

Operation

R0 ← R0 | Rn

PSR[2] ← R0[7]

PSR[0] ← (R0 == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	-	X

Clocks

1

Description

Bitwise OR the contents of a general purpose register with the contents of the accumulator, storing the resulting value to the accumulator.

Negative and Zero Flags are set to reflect the new value of R0.

AND - Bitwise AND Register to Accumulator

Format

Opcode					Reg		
0	0	1	0	0	X	X	X

Mnemonic

[label:] AND R*n*

Operation

R0 ← R0 & Rn

PSR[2] ← R0[7]

PSR[0] ← (R0 == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	-	X

Clocks

1

Description

Bitwise AND the contents of a general purpose register with the contents of the accumulator, storing the resulting value to the accumulator.

Negative and Zero Flags are set to reflect the new value of R0.

XOR - Bitwise XOR Register to Accumulator

Format

Opcode					Reg		
0	0	1	0	1	X	X	X

Mnemonic

[label:] XOR R*n*

Operation

R0 ← R0 ^ Rn

PSR[2] ← R0[7]

PSR[0] ← (R0 == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	-	X

Clocks

1

Description

Bitwise XOR the contents of a general purpose register with the contents of the accumulator, storing the resulting value to the accumulator.

Negative and Zero Flags are set to reflect the new value of R0.

ROL – Rotate Left Register through Carry

Format

Opcode					Reg		
0	0	1	1	0	X	X	X

Mnemonic

[label:] ROL R*n*

Operation

TMP ← Rn[7]

Rn ← (Rn << 1) | PSR[1]

PSR[1] ← TMP

PSR[2] ← Rn[7]

PSR[0] ← (Rn == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	X	X

Clocks

1

Description

Rotate the value general purpose register left one bit, with the low order bit taking the value of the carry flag, and the carry flag being updated to the former high order bit of the value. Place the resulting value in the same general purpose register.

Negative and Zero Flags are set to reflect the new value of R*n*. Carry Flag is changed as described, above.

ROR – Rotate Right Register through Carry

Format

Opcode					Reg		
0	0	1	1	1	X	X	X

Mnemonic

[label:] ROR R*n*

Operation

TMP ← Rn[0]

Rn ← (Rn >> 1) | (PSR[1] << 7)

PSR[1] ← TMP

PSR[2] ← Rn[7]

PSR[0] ← (Rn == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	X	X

Clocks

1

Description

Rotate the value general purpose register right one bit, with the high order bit taking the value of the carry flag, and the carry flag being updated to the former low order bit of the value. Place the resulting value in the same general purpose register.

Negative and Zero Flags are set to reflect the new value of R*n*. Carry Flag (PSR[1]) is changed as described, above.

DEC – Decrement Register

Format

Opcode					Reg		
0	1	0	0	0	X	X	X

Mnemonic

[label:] DEC R*n*

Operation

Rn ← Rn + 0xFF

PSR[2] ← Rn[7]

PSR[1] ← Rn[8]

PSR[0] ← (Rn == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	X	X

Clocks

1

Description

Decrement the value in a general purpose register.

Negative, Carry, and Zero Flags are set to reflect the new value of R*n*.

SBC – Subtract Register and Carry Bit from Accumulator

Format

Opcode					Reg		
0	1	0	0	1	X	X	X

Mnemonic

[label:] SBC R*n*

Operation

R0 ← R0 + (-Rn) + PSR[1]

PSR[2] ← R0[7]

PSR[1] ← R0[8]

PSR[0] ← (R0 == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	X	X

Clocks

1

Description

Arithmetically subtract a general purpose register and the current value of the carry bit from the accumulator, with result returned to the accumulator.

Negative, Carry, and Zero Flags are set to reflect the new value of R0.

ADD – Add Register to Accumulator

Format

Opcode					Reg		
0	1	0	1	0	X	X	X

Mnemonic

[label:] ADD R*n*

Operation

R0 ← R0 + Rn

PSR[2] ← R0[7]

PSR[1] ← R0[8]

PSR[0] ← (R0 == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	X	X

Clocks

1

Description

Arithmetically add the value from a register to the value in the accumulator, placing the result in the accumulator.

Negative, Carry, and Zero Flags are set to reflect the new value of R0.

STP – Set Bit in PSR

Format

Opcode					N		
0	1	0	1	1	X	X	X

Mnemonic

[label:] STP *n*

Operation

PSR[n] ← 1

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
*	*	*	*	*	*	*	*

Clocks

1

Description

Set a bit in the Program Status Register (PSR) to a value of '1'.

Any bit in the PSR may be set. Only the specified bit is affected.

BTT – Bit Test in Accumulator

Format

Opcode					N		
0	1	1	0	0	X	X	X

Mnemonic

[label:] BTT *n*

Operation

PSR[0] ← ~R0[n]

PSR[2] ← R0[7]

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	-	X

Clocks

1

Description

Test a bit in the accumulator, setting Zero Flag to reflect zero/non-zero.

The Negative flag reflects the value of R0. The Zero flag is set as described.

CLP – Clear Bit in PSR

Format

Opcode					N		
0	1	1	0	1	X	X	X

Mnemonic

[label:] CLP *n*

Operation

PSR[n] ← 0

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
*	*	*	*	*	*	*	*

Clocks

1

Description

Clear a bit in the Program Status Register (PSR) to a value of zero.

Any bit in the PSR may be cleared. Only the specified bit is affected.

T0X – Transfer Accumulator to Register

Format

Opcode					Reg		
0	1	1	1	0	X	X	X

Mnemonic

[label:] T0X R*n*

Operation

Rn ← R0

PSR[2] ← Rn[7]

PSR[0] ← (Rn == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	-	X

Clocks

1

Description

Copy the value from the accumulator to a register.

Negative and Zero Flags are set to reflect the new value of R*n*.

CMP – Compare Register to Accumulator

Format

Opcode					Reg		
0	1	1	1	1	X	X	X

Mnemonic

[label:] CMP R*n*

Operation

TMP ← R0 + (–Rn)

PSR[2] ← TMP[7]

PSR[1] ← TMP[8]

PSR[0] ← (TMP == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	X	X

Clocks

1

Description

Set the Flags to reflect the result of adding a negated register to the value in the accumulator.

The result is discarded, and neither the register nor the accumulator is changed.

Negative, Carry, and Zero Flags are set to reflect the result of the addition.

PSH – Push Register onto Stack

Format

Opcode					Reg		
1	0	0	0	0	X	X	X

Mnemonic

[label:] PSH R*n*

Operation

MEM[--STACK] ← Rn

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3

Description

Push the value of the register onto the stack.

The stack pointer is decremented before the value is written to memory.

Flags are unchanged by this operation.

POP – Pop Register from Stack

Format

Opcode					Reg		
1	0	0	0	1	X	X	X

Mnemonic

[label:] POP R*n*

Operation

Rn ←MEM[STACK++]

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3

Description

Pop the value from the top of the stack into the register.

The stack pointer is incremented after the value has been retrieved from memory.

Flags are unchanged by this operation.

BR0 – Relative Branch if PSR Bit is Zero

Format

Opcode					Bit Index			Signed Offset							
1	0	0	1	0	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] BR0 *n, signed_offset*

Operation

If (PSR[n] == 0) {

PROGRAM_COUNTER ← PROGRAM_COUNTER + SIGN_EXTEND(offset)

}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3

Description

Take a relative branch if the specified PSR bit is zero. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

BR1 – Relative Branch if PSR Bit is One

Format

Opcode					Bit Index			Signed Offset							
1	0	0	1	1	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] BR1 *n, signed_offset*

Operation

If (PSR[n] == 1) {

PROGRAM_COUNTER ← PROGRAM_COUNTER + SIGN_EXTEND(signed_offset)

}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3

Description

Take a relative branch if the specified PSR bit is '1'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

DBNZ – Decrement Register and Branch Not Zero

Format

Opcode					Reg			Signed Offset							
1	0	1	0	0	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] DBNZ R*n, signed_offset*

Operation

Rn ← Rn + 0xFF

PSR[2] ← Rn[7]

PSR[1] ← Rn[8]

PSR[0] ← (Rn == 0)

If (PSR[0] == 0) {

PROGRAM_COUNTER ← PROGRAM_COUNTER + SIGN_EXTEND(signed_offset)

}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	X	X

Clocks

3

Description

Decrement the value in a general purpose register. If the resulting value is not zero, take a relative branch, otherwise proceed to the next instruction.

Negative, Carry, and Zero Flags are set to reflect the new value of R*n*.

INT – Interrupt

Format

Opcode					N		
1	0	1	0	1	X	X	X

Mnemonic

[label:] INT *n*

Operation

```
if ((n == 0) || ((PSR[3] == '1') && (INTERRUPT_MASK_REGISTER[n] == '1'))) {
        MEM[--STACK] ← PSR
        PSR[3] ← 0
        MEM[--STACK] ← HI8(PROGRAM_COUNTER )
        MEM[--STACK] ← LO8(PROGRAM_COUNTER )
        PROGRAM_COUNTER ← {MEM[VECTOR_BASE+(n*2)+1]:VECTOR_BASE+(n*2)]}
}
```

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	X	-	-	-

Clocks

7

Description

If the interrupt is interrupt 0 (NMI), or interrupts are enabled and the interrupt mask for this interrupt is '1', then push the current PSR and PROGRAM_COUNTER register onto the stack, set the PROGRAM_COUNTER to the value of the *n*th vector in the interrupt vector table, and set the Interrupt Enable Flag to '0', disabling maskable interrupts.

Otherwise, do nothing.

MUL – Multiply R0 by Register

Format

Opcode					Reg		
1	0	1	1	0	X	X	X

Mnemonic

[label:] MUL *Rn*

Operation

*R1:R0 ← Rn * R0*

PSR[0] ← ((R0 == 0) && (R1 == 0))

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	X

Clocks

2

Description

Multiply, unsigned, the contents of the accumulator by the contents of *Rn*. The sixteen-bit result is placed in the register pair *{R1:R0}*

Zero Flag is changed to reflect the new value of *{R1:R0}*.

RSP – Reset Stack Pointer

Opcode					Select		
1	0	1	1	1	0	0	0

Mnemonic

[label:] RSP

Operation

STACK ← {R1:R0}

OR

STACK ← 0x007f

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

1

Description

Set the stack pointer to the contents of *{R1:R0}* if the processor core has been configured with *Allow_Stack_Address_Move* equal to '1', otherwise set the stack pointer to the default initial value of 0x007f.

Flags are unchanged by this operation.

Note

The Open8 instruction may differ from the original corresponding instruction of the V8 and ARClite processors if the processor core has been configured with *Allow_Stack_Address_Move* equal to '1'.

Additionally, different configurations of the Open8 may have differing behavior for this instruction.

Code intended to be portable should not use this instruction.

RTS – Return from Subroutine

Format

Opcode					Select		
1	0	1	1	1	0	0	1

Mnemonic

[label:] RTS

Operation

PROGRAM_COUNTER ← {MEM[STACK+1]:MEM[STACK]}

STACK ← STACK + 2

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

5

Description

Pop the address of the next instruction to execute from the stack.

Flags are unchanged by this operation.

RTI – Return from Interrupt

Format

Opcode					Select		
1	0	1	1	1	0	1	0

Mnemonic

[label:] RTI

Operation

PROGRAM_COUNTER ← {MEM[STACK+1] :MEM[STACK]}

PSR ← MEM[STACK+2]

STACK ← STACK + 3

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
X	X	X	X	X	X	X	X

Clocks

5

Description

Pop the address of the next instruction to execute from the stack, and restore the PSR from the stack.

All of the flags are replaced by this operation with the values saved on the stack by the prior interrupt. This may enable interrupts.

BRK - Break

Format

Opcode					Select		
1	0	1	1	1	0	1	1

Mnemonic

[label:] BRK

Operation

No Operation

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

1

Description

No operation is performed

Flags are unaffected by this instruction.

JMP – Jump to Absolute Address

Format

Opcode					Select			Address													
1	0	1	1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] JMP *unsigned_16_bit_address*

Operation

PROGRAM_COUNTER ← unsigned_16_bit_address

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3

Description

Continue execution at the memory address specified.

Flags are unaffected by this instruction.

SMSK – Set Interrupt Mask†

Format

Opcode					Select		
1	0	1	1	1	1	0	1

Mnemonic

[label:] SMSK

Operation

INTERRUPT_MASK_REGISTER ← (R0 | 0x01)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

1

Description

The interrupt mask register is set with the value of R0, or'ed with a 0x01. Effectively, the low order bit of the interrupt mask register, the enable bit for NMI, is always set to '1'.

Flags are unaffected by this instruction.

† Implemented on Open8, only.

GMSK – Get Interrupt Mask†

Format

Opcode					Select		
1	0	1	1	1	1	1	0

Mnemonic

[label:] GMSK

Operation

R0 ← INTERRUPT_MASK_REGISTER

PSR[2] ← R0[7]

PSR[0] ← 0

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	-	0

Clocks

1

Description

Register R0 is loaded with the current value of the interrupt mask register.

The Negative flag is set to reflect the new value of R0. The Zero flag is always set to '0', as the low order bit of the interrupt mask register (the enable mask for NMI) is always set to '1'.

† Implemented on Open8, only.

JSR – Jump to Subroutine

Format

Opcode					Select			Address													
1	0	1	1	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] JSR *unsigned_16_bit_address*

Operation

TMP ← PROGRAM_COUNTER + 3

MEM[--STACK] ← HI8(TMP)

MEM[--STACK] ← LO8(TMP)

PROGRAM_COUNTER ← unsigned_16_bit_address

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

5

Description

Push the address of the following instruction on the stack.

Continue execution at the memory address specified.

Flags are unaffected by this instruction.

UPP – Increment Register Pair

Format

Opcode					Reg		
1	1	0	0	0	X	X	0

Mnemonic

[label:] UPP *Re*

Operation

{Re+1:Re} ← {Re+1:Re} + 1

PSR[1] ← {Re+1:Re}[16]

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	X	-

Clocks

2

Description

Increment the 16-bit value contained in register pair *{Re+1:Re}*.

The Carry Flag is set to reflect a carry out of the high order bit of *{Re+1:Re}*.

STA – Store Register to Address

Format

Opcode					Reg			Address													
1	1	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] STA R*n, unsigned_16_bit_address*

Operation

MEM[unsigned_16_bit_address] ← Rn

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

4

Description

Store the contents of the specified register at the memory address.

Flags are unaffected by this instruction.

STX – Store R0 Indexed

Format

Opcode					Reg		
1	1	0	1	0	X	X	A

Mnemonic

[label:] STX *Re*[++]

Operation

MEM[{Re+1:Re}] ← R0

{Re+1:Re} ← {Re+1:Re} + A

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3

Description

Store the contents of the accumulator at the memory address specified by the register pair *{Re+1:Re}*.

The value of *{Re+1:Re}* is treated as an unsigned 16-bit value. The value of the *"A"* field is added to *{Re+1:Re}* each time the instruction is executed.

Flags are unaffected by this instruction.

Notes

If the auto-increment operator is specified , the low-order bit of the **Reg** field in the instruction (the *"A"* field) is set to '1'.

STO – Store R0 Indexed with Offset

Format

Opcode					Reg			Unsigned Offset							
1	1	0	1	1	X	X	A	X	X	X	X	X	X	X	X

Mnemonic

[label:] STO *Re*[++], *unsigned_offset*

Operation

MEM[{Re+1:Re} + unsigned_offset] ← R0

{Re+1:Re} ← {Re+1:Re} + A

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

4

Description

Store the contents of the accumulator at the memory address specified by *{Re+1:Re} + unsigned_offset.*

R*e* is restricted to even numbered registers.

The value of *{Re+1:Re}* is treated as an unsigned 16-bit value. The value of the *"A"* field is added to *{Re+1:Re}* each time the instruction is executed.

Flags are unaffected by this instruction.

Notes

If the auto-increment operator is specified , the low-order bit of the **Reg** field (the *"A"* field) in the instruction is set to '1'.

LDI – Load Register Immediate

Format

Opcode					Reg			Immediate Value							
1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] LDI R*n, immediate8*

Operation

Rn ← immediate8

PSR[2] ← Rn[7]

PSR[0] ← (Rn == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	-	X

Clocks

2

Description

The immediate 8-bit value is loaded into R*n*.

The Negative and Zero Flags are set to reflect the new value of R*n*.

Note

The assembler supports two functions that may appear in the expressions for the immediate value of LDI, "*hi8(expression)*" and "*lo8(expression)*". These builtin compile-time functions return the most significant 8 bits and the least significant 8 bits, respectively, of 16 bit expressions that can form addresses. This is convenient for setting register pairs for address operations.

LDA – Load Register from Address

Format

Opcode					Reg			Address													
1	1	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] LDA R*n, unsigned_16_bit_address*

Operation

Rn ← MEM[unsigned_16_bit_address]

PSR[2] ← Rn[7]

PSR[0] ← (Rn == 0)

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	-	X

Clocks

4

Description

Load the contents at the memory address into the specified register.

The Negative and Zero Flags are set to reflect the new value of Rn.

LDX – Load R0 Indexed

Format

Opcode					Reg		
1	1	1	1	0	X	X	A

Mnemonic

[label:] LDX *Re*[++]

Operation

R0 ← MEM[{Re+1:Re}]

PSR[2] ← R0[7]

PSR[0] ← (R0 == 0)

{Re+1:Re} ← {Re+1:Re} + A

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	-	X

Clocks

3

Description

Load the contents of the memory address specified by the register pair *{Re+1:Re}* into R0.

The value of *{Re+1:Re}* is treated as an unsigned 16-bit value. The value of the *"A"* field is added to *{Re+1:Re}* each time the instruction is executed.

The Negative and Zero Flags are set to reflect the new value of R0.

Notes

If the auto-increment operator is specified , the low-order bit of the Reg field in the instruction is set to '1'.

LDO - Load R0 Indexed with Offset

Format

Opcode					Reg			Unsigned Offset							
1	1	1	1	1	X	X	A	X	X	X	X	X	X	X	X

Mnemonic

[label:] LDO *Re*[++], *unsigned_offset*

Operation

R0 ← MEM[{Re+1:Re} + unsigned_offset]

PSR[2] ← R0[7]

PSR[0] ← (R0 == 0)

{Re+1:Re} ← {Re+1:Re} + A

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	X	-	X

Clocks

4

Description

Load the contents of the memory address specified by (*{Re+1:Re} + unsigned_offset)* into R0.

The value of *{Re+1:Re}* is treated as an unsigned 16-bit value. The value of the *"A"* field is added to *{Re+1:Re}* each time the instruction is executed.

Negative and Zero Flags are set to reflect the new value of R0.

Notes

If the auto-increment operator is specified , the low-order bit of the Reg field (the “auto-increment” or “A” field) in the instruction is set to '1'.

Instruction Pseudo-Mnemonics

The following instructions are aliases for particular hardware instructions, and are included primarily for programmer convenience.

STI - Set Interrupt Enable Flag

Format

Opcode					Reg		
0	1	0	1	1	0	1	1

Mnemonic

[label:] STI

Operation

PSR[3] ← 1

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	1	-	-	-

Clocks

1

Description

Sets the Interrupt Enable Flag in the Program Status Register (PSR) to a value of '1'.

Only the specified bit is affected.

Instruction Alias

[label:]STP 3

CLI – Clear Interrupt Enable Flag

Format

Opcode					Reg		
0	1	1	0	1	0	1	1

Mnemonic

[label:] CLI

Operation

PSR[3] ← 0

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	0	-	-	-

Clocks

1

Description

Clears the Interrupt Enable Flag in the Program Status Register (PSR) to a value of '0'.

Only the specified bit is affected.

Instruction Alias

[label:]CLP 3

STZ – Set Zero Flag

Format

Opcode					Reg		
0	1	0	1	1	0	0	0

Mnemonic

[label:] STZ

Operation

PSR[0] ← 1

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	1

Clocks

1

Description

Sets the Zero Flag in the Program Status Register (PSR) to a value of '1'.

Only the specified bit is affected.

Instruction Alias

[label:]STP 0

CLZ – Clear Zero Flag

Format

Opcode					Reg		
0	1	1	0	1	0	0	0

Mnemonic

[label:] CLZ

Operation

PSR[0] ← 0

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	0

Clocks

1

Description

Clears the Zero Flag in the Program Status Register (PSR) to a value of '0'.

Only the specified bit is affected.

Instruction Alias

[label:]CLP 0

STN – Set Negative Flag

Format

Opcode					Reg		
0	1	0	1	1	0	1	0

Mnemonic

[label:] STN

Operation

PSR[2] ← 1

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	1	-	-

Clocks

1

Description

Sets the Negative Flag in the Program Status Register (PSR) to a value of '1'.

Only the specified bit is affected.

Instruction Alias

[label:]STP 2

CLN – Clear Negative Flag

Format

Opcode					Reg		
0	1	1	0	1	0	1	0

Mnemonic

[label:] CLN

Operation

PSR[2] ← 0

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	0	-	-

Clocks

1

Description

Clears the Negative Flag in the Program Status Register (PSR) to a value of '0'.

Only the specified bit is affected.

Instruction Alias

[label:]CLP 2

STC – Set Carry Flag

Format

Opcode					Reg		
0	1	0	1	1	0	0	1

Mnemonic

[label:] STC

Operation

PSR[1] ← 1

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	1	-

Clocks

1

Description

Sets the Carry Flag in the Program Status Register (PSR) to a value of '1'.

Only the specified bit is affected.

Instruction Alias

[label:]STP 1

CLC – Clear Carry Flag

Format

Opcode					Reg		
0	1	1	0	1	0	0	1

Mnemonic

[label:] CLC

Operation

PSR[1] ← 0

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	0	-

Clocks

1

Description

Clears the Carry Flag in the Program Status Register (PSR) to a value of '0'.

Only the specified bit is affected.

Instruction Alias

[label:]CLP 1

NOP – No Operation

Format

Opcode					Select		
1	0	1	1	1	0	1	1

Mnemonic

[label:] NOP

Operation

No Operation

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

1

Description

Proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BRK

BRZ – Branch on Zero

Format

Opcode					Bit Index			Signed Offset							
1	0	0	1	1	0	0	0	X	X	X	X	X	X	X	X

Mnemonic

[label:] BRZ *signed_offset*

Operation

If (PSR[0] == 1) {

PROGRAM_COUNTER ← PROGRAM_COUNTER + SIGN_EXTEND(offset)

}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3

Description

Take a relative branch if the Zero Flag is '1'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BR1 0, signed_offset

BRNZ – Branch on Not Zero

Format

Opcode					Bit Index			Signed Offset							
1	0	0	1	0	0	0	0	X	X	X	X	X	X	X	X

Mnemonic

[label:] BRNZ *signed_offset*

Operation

If (PSR[0] == 0) {

PROGRAM_COUNTER ← PROGRAM_COUNTER + SIGN_EXTEND(offset)

}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag<	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3

Description

Take a relative branch if the Zero Flag is '0'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BR0 0, signed_offset

BRLZ – Branch on Less than Zero

Format

Opcode					Bit Index			Signed Offset							
1	0	0	1	1	0	1	0	X	X	X	X	X	X	X	X

Mnemonic

[label:] BRLZ *signed_offset*

Operation

If (PSR[2] == 1) {

PROGRAM_COUNTER ← PROGRAM_COUNTER + SIGN_EXTEND(offset)

}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3nn

Description

Take a relative branch if the Negative Flag is '1'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BR1 2, signed_offset

BRGEZ – Branch on Greater than or Equal to Zero

Format

Opcode					Bit Index			Signed Offset							
1	0	0	1	0	0	1	0	X	X	X	X	X	X	X	X

Mnemonic

[label:] BRGEZ *signed_offset*

Operation

If (PSR[2] == 0) {

PROGRAM_COUNTER ← PROGRAM_COUNTER + SIGN_EXTEND(offset)

}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3

Description

Take a relative branch if the Negative_Flag is '0'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BR0 2, signed_offset

BRC – Branch on Carry

Format

Opcode					Bit Index			Signed Offset							
1	0	0	1	1	0	0	1	X	X	X	X	X	X	X	X

Mnemonic

[label:] BRC *signed_offset*

Operation

If (PSR[1] == 1) {

PROGRAM_COUNTER ← PROGRAM_COUNTER + SIGN_EXTEND(offset)

}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3

Description

Take a relative branch if the Carry Flag is '1'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BR1 1, signed_offset

BRNC – Branch on Not Carry

Format

Opcode					Bit Index			Signed Offset							
1	0	0	1	0	0	0	1	X	X	X	X	X	X	X	X

Mnemonic

[label:] BRNC *signed_offset*

Operation

If (PSR[1] == 0) {

PROGRAM_COUNTER ← PROGRAM_COUNTER + SIGN_EXTEND(offset)

}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3

Description

Take a relative branch if the Carry Flag is '0'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BR0 1, signed_offset

JMPZ - Jump on Zero

Format

Opcode					Bit Index			Signed Offset								[redacted]
1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	[redacted]
Opcode					**Select**			**Address**								
1	0	1	1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] JMPZ *unsigned_16_bit_address*

Operation

If (PSR[0] == 0) {
PROGRAM_COUNTER ← PROGRAM_COUNTER + 5
} else {
PROGRAM_COUNTER ← unsigned_16_bit_address
}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3 or 6

Description

Jump to address if the Zero Flag is '1'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BRNZ .+5
JMP *signed_offset*

JMPNZ - Jump on Not Zero

Format

Opcode					Bit Index			Signed Offset								
1	0	0	1	1	0	0	0	0	0	0	0	0	1	0	1	[redacted]
Opcode					Select			Address								
1	0	1	1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] JMPNZ *unsigned_16_bit_address*

Operation

If (PSR[0] == 1) {
PROGRAM_COUNTER ← PROGRAM_COUNTER + 5
} else {
PROGRAM_COUNTER ← unsigned_16_bit_address
}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3 or 6

Description

Jump to address if the Zero Flag is '0'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BRZ .+5
JMP *signed_offset*

JMPLZ - Jump on Less than Zero

Format

Opcode					Bit Index			Signed Offset								[redacted]							
1	0	0	1	0	0	1	0	0	0	0	0	0	1	0	1	[redacted]							
Opcode					Select			Address															
1	0	1	1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] JMPLZ *unsigned_16_bit_address*

Operation

If (PSR[2] == 0) {
PROGRAM_COUNTER ← PROGRAM_COUNTER + 5
} else {
PROGRAM_COUNTER ← unsigned_16_bit_address
}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3 or 6

Description

Jump to address if the Negative Flag is '1'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BRGEZ .+5
JMP *signed_offset*

JMPGEZ - Jump on Greater than or Equal to Zero

Format

Opcode					Bit Index			Signed Offset								
1	0	0	1	1	0	1	0	0	0	0	0	0	1	0	1	[redacted]

Opcode					Select			Address													
1	0	1	1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] JMPGEZ *unsigned_16_bit_address*

Operation

If (PSR[2] == 1) {
PROGRAM_COUNTER ← PROGRAM_COUNTER + 5
} else {
PROGRAM_COUNTER ← unsigned_16_bit_address
}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3 or 6

Description

Jump to address if the Negative Flag is '0'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BRLZ .+5
JMP *signed_offset*

JMPC - Jump on Carry

Format

Opcode					Bit Index			Signed Offset								
1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1	
Opcode					**Select**			**Address**								
1	0	1	1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] JMPC *unsigned_16_bit_address*

Operation

If (PSR[1] == 0) {
PROGRAM_COUNTER ← PROGRAM_COUNTER + 5
} else {
PROGRAM_COUNTER ← unsigned_16_bit_address
}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3 or 6

Description

Jump to address if the Carry Flag is '1'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BRNC .+5
JMP *signed_offset*

JMPNC - Jump on Not Carry

Format

Opcode					Bit Index			Signed Offset								
1	0	0	1	1	0	0	1	0	0	0	0	0	1	0	1	[redacted]

Opcode					Select			Address													
1	0	1	1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Mnemonic

[label:] JMPNC *unsigned_16_bit_address*

Operation

If (PSR[1] == 1) {
PROGRAM_COUNTER ← PROGRAM_COUNTER + 5
} else {
PROGRAM_COUNTER ← unsigned_16_bit_address
}

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	-	-	-

Clocks

3 or 6

Description

Jump to address if the Carry Flag is '0'. Otherwise, proceed to the next instruction.

Flags are unaffected by this instruction.

Instruction Alias

[label:] BRC .+5
JMP *signed_offset*

CLR - Clear Accumulator

Format

Opcode					Reg		
0	0	1	0	1	0	0	0

Mnemonic

[label:] CLR

Operation

R0 ← 0

PSR[2] ← 0

PSR[0] ← 1

Flags

PSR[7]	PSR[6]	PSR[5]	PSR[4]	PSR[3]	PSR[2]	PSR[1]	PSR[0]
GP4_Flag	GP3_Flag	GP2_Flag	GP1_Flag	Interrupt_Enable	Negative_Flag	Carry_Flag	Zero_Flag
-	-	-	-	-	0	-	1

Clocks

1

Description

Clear the accumulator to zero.

Negative Flag is set to 0, and Zero Flags is set to 1.

Instruction Alias

[label:] XOR R0

Open8 Application Binary Interface

[To Be Written]

Appendix A - Differences Between the Open8 and the V8

The Open8 implementation does not switch register banks when an interrupt occurs, as there is no interrupt register bank implemented.

The Open8 implements the USR instruction of the V8/ARClite as the DBNZ – Decrement Register and Branch Not Zero instruction.

The Open8 implements the USR2 instruction as the MUL – Multiply R0 by Register instruction.

The Open8 optionally implements autoincrement of the register address for the STX, STO, LDX, and LDO instructions.

The Open8 implements the SMSK – Set Interrupt Mask and GMSK – Get Interrupt Mask instructions.

Appendix B - Instruction Opcode Map

Instructions are listed in sorted hexadecimal order.

Hex Opcode Range	Instruction Mnemonic	Length/ Clocks	Instruction Reference
0x00-0x07	INC	1/1	INC – Increment Register
0x08-0x0F	ADC	1/1	ADC – Add Register and Carry Flag to Accumulator
0x10-0x17	TX0	1/1	TX0 – Transfer Register to Accumulator
0x18-0x1F	OR	1/1	OR – Bitwise OR Register to Accumulator
0x20-0x27	AND	1/1	AND - Bitwise AND Register to Accumulator
0x28-0x2F	XOR	1/1	XOR - Bitwise XOR Register to Accumulator
0x30-0x37	ROL	1/1	ROL – Rotate Left Register through Carry
0x38-0x3F	ROR	1/1	ROR – Rotate Right Register through Carry
0x40-0x47	DEC	1/1	DEC – Decrement Register
0x48-0x4F	SBC	1/1	SBC – Subtract Register and Carry Bit from Accumulator
0x50-0x57	ADD	1/1	ADD – Add Register to Accumulator
0x58-0x5F	STP	1/1	STP – Set Bit in PSR
0x60-0x67	BTT	1/1	BTT – Bit Test in Accumulator
0x68-0x6F	CLP	1/1	CLP – Clear Bit in PSR
0x70-0x77	T0X	1/1	T0X – Transfer Accumulator to Register
0x78-0x7F	CMP	1/1	CMP – Compare Register to Accumulator
0x80-0x87	PSH	1/3	PSH – Push Register onto Stack
0x88-0x8F	POP	1/3	POP – Pop Register from Stack
0x90-0x97	BR0	2/3	BR0 – Relative Branch if PSR Bit is Zero
0x98-0x9F	BR1	2/3	BR1 – Relative Branch if PSR Bit is One
0xA0-0xA7	DBNZ[1]	2/3	DBNZ – Decrement Register and Branch Not Zero
0xA8-0xAF	INT	1/?	INT – Interrupt
0xB0-0xB7	MUL[2]	1/2	MUL – Multiply R0 by Register
0xB8	RSP	1/1	RSP – Reset Stack Pointer
0xB9	RTS	1/5	RTS – Return from Subroutine
0xBA	RTI	1/5	RTI – Return from Interrupt

1 This opcode is mapped to USR on the V8/ARClite processor cores.
2 This opcode is mapped to USR2 on the V8/ARClite processor cores.

Hex Opcode Range	Instruction Mnemonic	Length/ Clocks	Instruction Reference
0xBB	BRK	1/1	BRK - Break
0xBC	JMP	3/3	JMP – Jump to Absolute Address
0xBD	SMSK[3]	1/1	SMSK – Set Interrupt Mask
0xBE	GMSK[4]	1/1	GMSK – Get Interrupt Mask
0xBF	JSR	3/5	JSR – Jump to Subroutine
0xC0-0xC6[5]	UPP	1/2	UPP – Increment Register Pair
0xC1-0xC7[6]	unassigned	1/1	
0xC8-0xCF	STA	3/4	STA – Store Register to Address
0xD0-0xD7	STX	1/3	STX – Store R0 Indexed
0xD8-0xDF	STO	2/4	STO – Store R0 Indexed with Offset
0xE0-0xE7	LDI	2/2	LDI – Load Register Immediate
0xE8-0xEF	LDA	3/4	LDA – Load Register from Address
0xF0-0xF8	LDX	1/3	LDX – Load R0 Indexed
0xF8-0xFF	LDO	2/4	LDO - Load R0 Indexed with Offset

3 Differs from V8/ARClite, which has this opcode unassigned.
4 Differs from V8/ARClite, which has this opcode unassigned.
5 Even opcodes only – the low order bit of the opcode is required to be '0', as only even numbered registers are allowed in the subop field.
6 Odd opcodes only.