Instruction Set Architecture

Project RISCII (PR0001)

“Interface between RISCII processor and compiled software”

Rev A
23-JUN-2024

Purpose

This document is meant to describe how the RISCII hardware and software “interact” with each other. It primarily defines how hardware interprets bits into executable software instructions, but also includes related details (e.g. how binary images should be formatted).

The contents of this document should effectively describe the high-level interface of the processor and low-level interface of the compiler. It is expected that this document is referenced in developing both systems.

Abbreviations

Abbreviation	Description
ISA	Instruction Set Architecture

References

Name	Document Number
-	-

Instruction Set

This section describes the core of the ISA: the instruction set. Instructions are 16-bit values that directly control the processor hardware. By sequencing instructions properly, entire functions and programs can be created.

Notable Registers

The following table describes registers (i.e. memory blocks) referenced or affected by the instruction set. These names are referenced further in the document.

Register	**Size**	**Description**
PC	16 bits	Program Counter- program address of instruction being read
R0 - R7	8 registers, 16 bits each	Register File- location of operands used by most instructions
CC	4 bits	Condition Codes- stores information about last computed value. This information is stored as flags “nzpc”
MEM	32768 registers, 16 bits each	Memory- data address space reserved for data and memory-mapped internal peripherals (distinct from program address space referenced by program counter (PC))
PCsh	16 bits	Shadow Program Counter- saves program counter (PC) value in special scenarios
CCsh	4 bits	Shadow Condition Codes- saves condition codes (CC) value in special scenarios
IL	1 bit	Interrupt Latch- saves whether or not the hardware is currently servicing an interrupt routine
HL	1 bit	Halt Latch- saves whether or not the hardware is in a permanent halted state

- Table 1: Instruction affected registers

Instructions

The following subsections explain each instruction in detail. It should be kept in mind that all operations are done with 16-bit integer precision. Likewise, all address accesses are done with 16-bit precision (i.e. word addressable, not byte addressable- though addresses assume such for software consistency).

For a summary of each instruction, see Appendix A: Instruction Set Summary.

AND: Bitwise AND

Formats:
[1111] [DST] [SR1] [000] [SR2]
[1111] [DST] [SR1] [1] [5bImm]

Performs a bitwise AND operation between SR1 and either SR2 or the encoded, sign-extended 5-bit value “5bImm”. Result is stored in DST.

SR1, SR2, and DST are registers within the register file (R0 - R7). Condition Codes (CC) are updated by this instruction.

ORR: Bitwise OR

Formats:
[1110] [DST] [SR1] [000] [SR2]
[1110] [DST] [SR1] [1] [5bImm]

Performs a bitwise OR operation between SR1 and either SR2 or the encoded, sign-extended 5-bit value “5bImm”. Result is stored in DST.

SR1, SR2, and DST are registers within the register file (R0 - R7). Condition Codes (CC) are updated by this instruction.

XOR: Bitwise XOR

Formats:
[1011] [DST] [SR1] [000] [SR2]
[1011] [DST] [SR1] [1] [5bImm]

Performs a bitwise XOR operation between SR1 and either SR2 or the encoded, sign-extended 5-bit value “5bImm”. Result is stored in DST.

SR1, SR2, and DST are registers within the register file (R0 - R7). Condition Codes (CC) are updated by this instruction.

SHL: Shift Logical Left

Formats:
[1100] [DST] [SR1] [000] [SR2]
[1100] [DST] [SR1] [10] [4bIm]

Performs shift logical left operation of SR1 by either SR2 or the encoded, zero-extended 4-bit value “4bIm”. Result is stored in DST.

SR1, SR2, and DST are registers within the register file (R0 - R7). Condition Codes (CC) are updated by this instruction.

SHR: Shift Right

Formats:
[1101] [DST] [SR1] [0] [a] [0] [SR2]
[1101] [DST] [SR1] [1] [a] [4bImm]

Performs shift right operation of SR1 by either SR2 or the encoded, zero-extended 4-bit value “4bImm”. Result is stored in DST.

Flag “a” indicates the type of shift executed. If the flag is set, a shift arithmetic right is performed. Otherwise, a shift logical right is performed.

SR1, SR2, and DST are registers within the register file (R0 - R7). Condition Codes (CC) are updated by this instruction.

ADD: Arithmetic Addition

Formats:
[1000] [DST] [SR1] [000] [SR2]
[1000] [DST] [SR1] [1] [5bImm]

Performs arithmetic addition operation of SR1 and either SR2 or the encoded, sign-extended 5-bit value “5bImm”. Result is stored in DST.

SR1, SR2, and DST are registers within the register file (R0 - R7). Condition Codes (CC) are updated by this instruction.

SUB: Arithmetic Subtraction

Formats:
[1001] [DST] [SR1] [000] [SR2]
[1001] [DST] [SR1] [1] [5bImm]

Performs arithmetic subtraction operation of SR1 from either SR2 or the encoded, sign-extended 5-bit value “5bImm” (i.e. SR1 is subtracted from 2nd operand). Result is stored in DST.

SR1, SR2, and DST are registers within the register file (R0 - R7). Condition Codes (CC) are updated by this instruction.

LBI: Load Byte Immediate

Formats:
[1010] [DST] [s] [8bImmVal]

Sets DST to either the encoded, sign-extended 8-bit value “8bImmVal” or “shifted” variant of DST. “Shifted” variant is equal to DST logical left shifted by the 8, then OR-ed with “8bImmVal” (i.e. “8bImmVal” is left shifted into DST).

Flag “s” indicates which value is saved to DST. If the flag is set, the encoded, sign-extended 16-value is saved. Otherwise, the “shifted” variant is saved.

DST is a register within the register file (R0 - R7). Condition Codes (CC) are updated by the instruction.

LDR: Load Memory Word

Formats:
[0101] [DST] [SR1] [6bOfst]

Loads the two-byte word value stored at a computed memory location in MEM and saves it to DST. Memory address accessed is equivalent to the encoded, sign-extended 6-bit value “6bOfst” logical left shifted by 1, then summed with SR1 (i.e. pointer offset addressing).

The computed address always refers to the data address space (i.e. MEM). Despite the address being byte addressable, the instruction always reads two bytes- floored to the nearest word aligned address.

SR1 and DST are registers within the register file (R0 - R7). Condition Codes (CC) are NOT updated by this instruction.

STR: Store Memory Word

Formats:
[0100] [SRC] [SR1] [6bOfst]

Stores the two-byte word value in SRC to a computed memory location in MEM. Memory address accessed is equivalent to the encoded, sign-extended 6-bit value “6bOfst” logical left shifted by 1, then summed with SR1 (i.e. pointer offset addressing).

The computed address always refers to the data address space (i.e. MEM). Despite the address being byte addressable, the instruction always reads two bytes- floored to the nearest word aligned address.

SRC and DST are registers within the register file (R0 - R7). Condition Codes (CC) are NOT updated by this instruction.

BRC: Branch Conditional

Formats:
[0010] [nzpc] [8bImmVal]

Jump to instruction relative to PC + 2 address if certain conditions are met. The new PC value, if conditions are met, is equivalent to the encoded, sign-extended 8-bit value “8bImmVal” logical left shifted by 1, then summed with PC + 2. Otherwise, the instruction has no explicit effect on the PC.

The following conditions must be met in order to explicitly update the PC:

1) At least one of the instruction’s “nzp” flags matches its corresponding CC flag
2) If the instruction’s “c” flag is set, the corresponding CC flag is also set

In general, setting a flag enables the condition to potentially update the PC. Clearing a flag prevents the condition, even if true, from updating the PC.

Each flag implies information about the last instruction that updated the CC register/flags. See Appendix B: Condition Flag Information for more information.

Condition Codes (CC) are NOT updated by this instruction.

JPR: Jump Base Register

Formats:
[0110] [000] [SR1] [r] [5bImm]

Sets PC to a computed memory location. New, computed PC address is equivalent to the encoded, sign-extended 5-bit value “5bImm” logical left shifted by 1, then summed with SR1 (i.e. pointer offset addressing).

The new, computed PC address always refers to the program address space. Despite the address being byte addressable, the address is always floored to the nearest word aligned address.

Flag “r” rather significantly changes the instruction’s behavior (though is semantically similar). If the flag is set, the instruction’s other arguments are ignored and instead updates both PC and CC with the values of PCsh and CCsh, respectively. Also, IL latch is cleared. Otherwise, if the flag is cleared, the behavior described above is executed.

SR1 is a register within the register file (R0 - R7). Condition Codes (CC) are NOT updated by this instruction.

JLR: Jump and Link Register

Formats:
[0111] [DST] [SR1] [0] [5bImm]

Sets the PC to a computed memory location while saving the PC + 2 to DST. New, computed PC address is equivalent to the encoded, sign-extended 5-bit value “5bImm” logical left shifted by 1, then summed with SR1 (i.e. pointer offset addressing).

The new, computed PC address always refers to the program address space. Despite the address being byte addressable, the address is always floored to the nearest word aligned address.

SRC and DST are registers within the register file (R0 - R7). Condition Codes (CC) are NOT updated by this instruction.

SWP: Swap Register and Memory

Formats:
[0001] [DST] [SR1] [6bOfst]

Swaps the values located in DST and a computed memory location in MEM. Memory address accessed is equivalent to the encoded, sign-extended 6-bit value “6bOfst” logical left shifted by 1, then summed with SR1 (i.e. pointer offset addressing). The instruction is performed as one, atomic operation.

The computed address always refers to the data address space (i.e. MEM). Despite the address being byte addressable, the instruction always reads two bytes- floored to the nearest word aligned address.

SRC and DST are registers within the register file (R0 - R7). Condition Codes (CC) are NOT updated by this instruction.

NOP: No Operation

Formats:
[0000] [000000000000]

This instruction performs no noticeable changes to any registers (besides typical incrementing of PC while executing instructions).

Condition Codes (CC) are NOT updated by this instruction.

HLT: Halt

Formats:
[0011] [000000000000]

Sets HL latch and prevents PC from being updated. Instruction prevents further instructions from being executed- either directly by the core processor or indirectly through interrupts/peripherals.

The only way to “move past” this instruction is a hardware level reset (e.g. power cycling the hardware).

Condition Codes (CC) are NOT updated by this instruction.

Memory Layout

This section describes the expected sizes and address spaces of the ISA. In general, it is expected that each address space is completely separate (i.e. do not overlap with each other) and are word addressable.

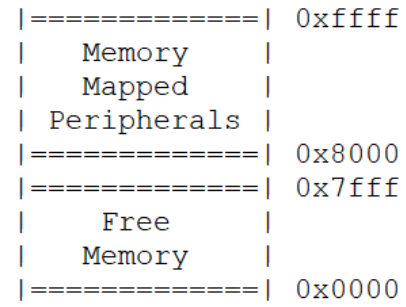
Program Address Space

The program address space is a 64 kilobyte section where instructions are stored during runtime. It is generally read only and addressed by the program counter (PC).

Data Address Space

The data address space (also referred to as MEM) is a 64 kilobyte section where data and memory-mapped internal peripherals can be accessed. It has a mix of read/write permissions and is accessed through instructions like LDR, STR, and SWP.

The lower half of the section is free memory for the program to use for data (e.g. stack and heap) while the upper half is memory-mapped to the hardware’s internal peripherals. The below image illustrates the two subsections within the address space.



- Image 1: Data Address Space Layout, byte addressed

Storage Address Space

The storage address space is a 64 kilobyte section where the binary image is stored in-between power cycles. It is not directly accessible to the program.

While the ISA does not specify how the binary image is formatted or how its contents reach the other address spaces, it is assumed the hardware copies at least the instructions to the program address space.

Appendix A: Instruction Set Summary

Instruction	**Format**	**Operation**
AND	[1111] [DST] [SR1] [000] [SR2] [1111] [DST] [SR1] [1] [5bImm]	Bitwise AND Register or immediate 2nd operand Updates CC
ORR	[1110] [DST] [SR1] [000] [SR2] [1110] [DST] [SR1] [1] [5bImm]	Bitwise OR Register or immediate 2nd operand Updates CC
XOR	[1011] [DST] [SR1] [000] [SR2] [1011] [DST] [SR1] [1] [5bImm]	Bitwise XOR Register or immediate 2nd operand Updates CC
SHL	[1100] [DST] [SR1] [000] [SR2] [1100] [DST] [SR1] [10] [4bIm]	Shift Logical Left Register or immediate 2nd operand Updates CC
SHR	[1101] [DST] [SR1] [0] [a] [0] [SR2] [1101] [DST] [SR1] [1] [a] [4bIm]	Shift Right Arithmetic or logical shift type Register or immediate 2nd operand Updates CC
ADD	[1000] [DST] [SR1] [000] [SR2] [1000] [DST] [SR1] [1] [5bImm]	Arithmetic Addition Register or immediate 2nd operand Updates CC
SUB	[1001] [DST] [SR1] [000] [SR2] [1001] [DST] [SR1] [1] [5bImm]	Arithmetic Subtraction Register or immediate 2nd operand SR1 is subtracted value Updates CC
LBI	[1010] [DST] [s] [8bImmVal]	Load Byte Immediate Sign extend or left shift in options

		Updates CC
LDR	[0101] [DST] [SR1] [6bOfst]	Load Memory Word 6bOfst is left shifted by 1
STR	[0100] [SRC] [SR1] [6bOfst]	Store Memory Word 6bOfst is left shifted by 1
BRC	[0010] [nzpc] [8bImmVal]	Branch Conditional Jumps relative to PC + 2 8bImmVal is left shifted by 1
JPR	[0110] [000] [SR1] [r] [5bImm]	Jump Base Register “Flag” sets PC/CC with PCsh/CCsh 5bImm is left shifted by 1
JLR	[0111] [DST] [SR1] [0] [5bImm]	Jump and Link Register 5bImm is left shifted by 1
SWP	[0001] [DST] [SR1] [6bOfst]	Swap Register and Memory 6bOfst is left shifted by 1
NOP	[0000] [000000000000]	No Operation
HLT	[0011] [000000000000]	Halt Sets HL to 1

- Table A1: Instruction Set Summary

Appendix B: Condition Flag Information

Condition Name	**Associated Flag**	**Condition If Set**
Negative	n	Jump if result is negative (as a signed integer)
Zero	z	Jump if result is zero
Positive	p	Jump if result is positive (as a signed integer)
Carry-out	c	Jump if result caused overflow/carry-out

- Table B1: Condition Flag Descriptions

Condition Flags	**Prior Operation (Before BRC)**	**Abstracted Meaning**
n	B - A	A > B (signed integers)

p	B - A	A < B (signed integers)
nz	B - A	A >= B (signed integers)
zp	B - A	A <= B (signed integers)
npc	A - B	A > B (unsigned integers)
npc	B - A	A < B (unsigned integers)
nzpc	A - B	A >= B (unsigned integers)
nzpc	B - A	A <= B (unsigned integers)
z	B - A	A == B
np	B - A	A != B
nzpc	<any operation>	carry-out occurred
nzp	<any operation>	PC-relative jump

- Table B2: Typical Condition Flag Combinations

Revision History

Rev	Date	Description of Changes	Initials
A	23-JUN-2024	Initial draft- focus on instruction set + mem layout	J.E.