Register Oriented Computer Architecture (ROC) Description

James Brakefield ©2020

Preface:

ROCA is part of a family of computer architectures. There are commonalities within the family: Little Endian, A register file, typically of 16 or 32 registers, and a main memory. The register bit length defines the largest data size and the maximum addressable main memory. Each family member supports three or four data sizes. The smallest addressable memory unit is called a byte and can be eight, nine or twelve bits in size. Distinct instruction sets exist for each member and each instruction set provides instructions of several sizes.

Immediate and displacement values needed by an instruction have a byte length field of three bits within a five bit register field and the immediate value follows the instruction (for the small values of -8 to 15, the value is coded within the instruction).

The family currently includes a RISC machine (ROC32_8), a stack/accumulator machine (STAOC32_8) and the multi-denominational ALT-A machine (combining stack, RISC and x86 style ISAs within a single architecture). Closely related are alternative variations on the MSP430, the VAX and the x86 (alt_msp430, alt_vax and alt_x86) ISAs.

Introduction:

ROCA has several variants that could be implemented. Byte size could be 12-bits leading to possible data sizes of 12, 24, 36 and 48 bits. More commonly byte size is 8-bits and the four data sizes are 8, 16, 24 and 32 bits; or 8, 16, 32 and 64 bits. The nomenclatures for these variations are: ROC48_12, ROC32_8 or ROC64_8. Herein ROC64_8 will be described.

ROC64_8 has four instruction sizes of 24, 32, 40 and 48 bits with any immediate value following the instruction. An immediate can be of any size from one to eight bytes. In general, instructions are byte aligned, however by choosing a suitable combination of instruction and immediate lengths, two or four byte alignment can be maintained.

ROCA has a 32 entry register file. Register zero is used to store the return address on subroutine entry. When used as an index register in load/store instructions it reads as zero. The program counter (PC) is available in register 31 ($1F), register 28 is the default location for the condition code register, and by convention, register 29 is the frame pointer and 30 is the stack pointer. Typically subroutine arguments are passed in the low numbered registers and results returned in their place.

The ROCA ISA has four instruction sizes and each larger size supports additional features with the register fields retaining their locations in larger sizes. The first two bits of the instruction indicate the instruction length. The remaining six bits of the first byte are the op-code.

The three byte long instructions take two operands from the register file and return one result. There is one bit used to indicate the second operand is an immediate. When second operand register field is an immediate, the five bit register field indicates a value from -8 to 15 or that a one to eight byte immediate follows the instruction. The three byte long load and store instructions are either base plus data size scaled index or base plus immediate displacement.

The four byte long instructions add a predicate generate bit that updates a condition code register and add a five bit condition select field to cause the instruction to conditionally execute. The remaining two bits of the fourth instruction byte are used to individually negate or invert (binary complement) the two operands. When the second operand is an immediate the second negate-invert bit becomes the operand swap enable.

Four byte long load and store instructions depart from the above: add an additional register field to the three byte long load and store instructions. Add a two bit index register scale factor field and a condition code update enable bit. The four byte store instructions replace the condition code update enable with a store immediate enable, so the store immediate instructions can have two independent immediate specifications: the address displacement and the value to be stored.

<u>Instruction Formats:</u>

Three byte with two operands	`00xxxxxx ddddd i rrrrr sssss`
Three byte with one operand	`00xxxxxx ddddd z rrrrr xxxxx`
Three byte with immediate	`00xxxxxx ddddd 1 rrrrr nnnnn`
Three byte load/store with index register	`00xxxxxx ddddd 0 bbbbb jjjjj`
Three byte load/store with displacement	`00xxxxxx ddddd 1 bbbbb nnnnn`
Four byte, two operands with predication	`01xxxxxx ddddd i rrrrr sssss zz u ccccc`
R op S ->D; execute on predicate match	`01xxxxxx ddddd 0 rrrrr sssss zz u ccccc`
Four byte, three operands	`01xxxxxx ddddd i rrrrr sssss zz u ttttt`
Four byte with immediate	`01xxxxxx ddddd 1 rrrrr nnnnn z v u ccccc`
4 byte load/store	`01xxxxxx ddddd i bbbbb sssss yy u jjjjj`
4 byte store immediate	`01xxxxxx mmmmm i bbbbb sssss yy 1 jjjjj`
5 byte with 3 source registers	`10xxxxxx ddddd i rrrrr sssss zzuccccc xxxttttt`
Six byte with 3 source registers, predication and 2 destination registers	`11xxxxxx ddddd i rrrrr sssss zzuccccc xxxttttt xxxeeeee`
Six byte with 4 source registers, predication and 2 destination registers	`11xxxxxx ddddd i rrrrr sssss zzuccccc xwwttttt wwweeeee`

<u>Bit field codes:</u>

00, 01, 10 or 11 Instruction length bits

B	Base address register
C	Five predicate condition bits (if condition matches, instruction is executed)
D	Destination or result register, exception: store instructions
E	Second destination or result register, replaces My 66000 Carry mechanism
I	Immediate flag bit, enables S to be a short constant (N) or an immediate length specification

ISZ	Part of the immediate bits (N), used to indicate number of following bytes of immediate
J	Index register, register zero reads as zero
M	Value/specification for store immediate value
N	Immediate or offset field or byte length of suffixed immediate
T	Third source register, for My 66000 Carry mechanism or three operand instructions
X	op-code bits
U	Update condition code register, register 28
V	Swap operand with immediate
W	Fourth source register
Y	Index register scaling: Data-size * (1, 2, 3 or 4)
Z	Invert/negate operand, one bit per register operand