Sonne Instruction Set Architecture

Oct. 2018 Michael Mangelsdorf ([mim@ok-schalter.de](mailto:mim@ok-schalter.de))

Memory

Sonne uses a 16-bit wide, 16-bit deep main memory (M) and a 16-bit wide frame memory (F).

The frame memory stores two structures: a stack of subroutine frames, and a conventional stack of memory cells. Each subroutine frame consists of 11 memory cells which are used as registers: 1 default register DR, 7 local registers L1-L7, one return address register R, one frame key register FK and one working register W used for code threading.

A subroutine call pushes a new subroutine frame onto the subroutine frame stack, a return from subroutine removes the current frame from the stack and resumes the previous subroutine frame. The push and pop operations use stack pointers that are not exposed to the programmer using base instructions.

A subroutine call instruction encodes four 3-bit offsets forming the 12-bit frame key. Each offset picks out one register in the caller frame that the called subroutine can use to receive values from and transparently update registers in the caller frame. Associated with these four offset are the four alias registers (A1-A4). When used by the caller, read and write accesses to these registers effectively operate on the corresponding registers in the caller frame.

The remaining four registers are two general purpose registers, JAR (Just A Register) and NEW, and two registers, 1ST and 2ND, mapped to the top two elements of the cell stack.

Instruction Format

Each instruction is 16-bits wide and its high-order 4 bits are the guide code G of the instruction. There are 24 base instructions, and in order to encode them, 1 additional bit (R2\_MSB) of the instruction word is alternatively used as a complement to the guide code, or as a significant bit as part of an operand (R2). To obtain the final operation code of the instruction, G is shifted up by 1-bit and then R2\_MSB is copied into the now empty least significant position, resulting in a 32 bit operation code. For instructions that require the entire range of the R2 operand, and for which R2\_MSB can therefore assume both its values, two operation codes map to the same instruction.

Instruction Set and Extensions

The 24 base instructions include three instructions for extending the instruction set: ZOP (Zero Operand), SOP (Single Operand) and DOP (Dual Operand), for which zero, one or two of the operand parts are used as operands, and the remaining ones are used to encode extended operation codes.

Phos:

// Decode instruction

G = (iw & 0xF000) >> 12; // Slot 0 - Guide code

L = (iw & 0xF00) >> 8; // Slot 1 - Left operand

R2 = (iw & 0xF0) >> 4; // Slot 2 - First right operand

R2\_MSB = (R2 & 8) >> 3; // Slot 2 - Most significant bit

R2\_LOR = R2 & 7; // Slot 3 - Remaining three bits

R1 = iw & 0xF; // Slot 3 - Second right operand

SEVEN = (R2\_LOR<<4) + R1;

OFFS = R2\_LOR + \*ref(R1);

SXOFFS = sxt(R2\_LOR,3) + \*ref(R1);

Paver:

`define G iw[15:12]

`define L iw[11:8]

`define R2 iw[7:4]

`define R1 iw[3:0]

`define R2\_MSB iw[7:7]

`define R2\_LOR iw[6:4]

`define SEVEN iw[6:0]

`define OFFS fq\_b + iw[6:4]

`define SXOFFS fq\_b + {{13{iw[6]}},iw[6:4]}

task op\_ZOP;

case ((`L<<4) | `R1)

task op\_SOP;

case( `SEVEN )

task op\_DOP;

case (`R2)