
SPECT

ISA v0.1

Version: 0.1

Git tag:

Tropic Square
September 13, 2023



1 Glossary

- $P_{25519} = 2^{255} - 19$
- $P_{256} = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$
- $||$ – concatenation

2 Instruction set

SPECT provides 4 types of instructions:

- **R** - Register
- **I** - Immediate
- **M** - Memory
- **J** - Jump

2.1 Operand interpretation

All operands are considered as 256 bits unsigneds. Arithmetic instructions working only with 32 bit operands ignores the 224 MSBs of input and clears them in the result. Logic instructions working only with 32 bit operands also ignores the 224 MSBs of input, but passes the 224 MSBs of op2 to the result.

2.2 Instruction Format

31	30	29	28	25	24	22	21	17	16	15	12	11	07	06	00	
	type	opcode	func				op1			op2			op3			R
	type	opcode	func				op1			op2			Immediate			I
	type	opcode	func				op1						Addr			M
	type	opcode	func										NewPC			J

2.3 Symbols

Following symbols are used in description of instructions:

- **F** – Flags set by the instruction
- **#C** – Number of cycles the instruction takes to execute



2.4 R instructions

Mnemonic	Name	Semantics	F	#C
32 bit arithmetic instructions				
ADD op1,op2,op3	32 bit addition	$op1 = op2 + op3$	Z	11
SUB op1,op2,op3	32 bit subtraction	$op1 = op2 - op3$	Z	11
CMP op2,op3	32 bit comparison	$op2 - op3$	Z	9
32 bit logic instructions				
AND op1,op2,op3	32 bit bitwise AND	$op1 = op2 \& op3$	Z	11
OR op1,op2,op3	32 bit bitwise OR	$op1 = op2 \mid op3$	Z	11
XOR op1,op2,op3	32 bit bitwise Exclusive OR	$op1 = op2 \wedge op3$	Z	11
NOT op1,op2	32 bit bitwise NOT	$op1 = \sim op2$	Z	10
Shift Instructions				
LSL op1,op2	Logic shift left	$op1 = op2[254:0] \mid \mid 0$	C	10
LSR op1,op2	Logic shift right	$op1 = 0 \mid \mid op2[255:1]$	C	10
ROL op1,op2	Rotating shift left	$op1 = op2[254:0] \mid \mid op2[255]$	C	10
ROR op1,op2	Rotating shift right	$op1 = op2[0] \mid \mid op2[255:1]$	C	10
ROL8 op1,op2	Rotating byte shift left	$op1 = op2[247:0] \mid \mid op2[255:248]$		10
ROR8 op1,op2	Rotating byte shift right	$op1 = op2[7:0] \mid \mid op2[255:8]$		10
SWE op1,op2	Swap endianity	$op1[255:248] = op2[7:0]$ $op1[247:240] = op2[15:8]$... $op1[7:0] = op2[255:248]$		10
Modular arithmetic instructions				
MUL25519 op1,op2,op3	Multiplication in $GF(P_{25519})$	$op1 = (op2 * op3) \% P_{25519}$		91
MUL256 op1,op2,op3	Multiplication in $GF(P_{256})$	$op1 = (op2 * op3) \% P_{256}$		139
ADDP op1,op2,op3	Generic Modular Addition	$op1 = (op2 + op3) \% R31$		16
SUBP op1,op2,op3	Generic Modular Subtraction	$op1 = (op2 - op3) \% R31$		16



Mnemonic	Name	Semantics	F	#C
MULP op1,op2,op3	Generic Modular Multiplication	$op1 = (op2 * op3) \% R31$		597
REDP op1,op2,op3	Generic Modular Reduction	$op1 = (op2 \mid \mid op3) \% R31$		528
Other Instructions				
MOV op1,op2	Move register	$op1 = op2$		7
CSWAP op1,op2	Conditional swap	if C == 1 then: $op1 = op2$ $op2 = op1$		11
HASH op1,op2	Hash	$tmp = SHA512(op2+3 \mid \mid op2+2 \mid \mid op2+1 \mid \mid op2)$ $op1 = tmp[255:0]$ $op1+1 = tmp[511:256]$		347
GRV op1	Get Random Value	$op1 = \text{Random number}$		-
SCB op1,op2,op3	Blind scalar	$B = \text{Blind}(op2, op3, R31)$ $op1 = B[255:0]$ $op1+1 = B[511:256]$		88

2.5 I instructions

Mnemonic	Name	Semantics	F	#C
32 bit arithmetic instructions				
ADDI op1,op2,Immediate	32 bit addition	$op1 = op2 + \text{Immediate}$	Z	11
SUBI op1,op2,Immediate	32 bit subtraction	$op1 = op2 - \text{Immediate}$	Z	11
CMPI op2,Immediate	32 bit comparison	$op2 - \text{Immediate}$	Z	9
12 bit logic instructions				
ANDI op1,op2,Immediate	12 bit bitwise logic AND	$op1 = op2 \& \text{Immediate}$	Z	11
ORI op1,op2,Immediate	12 bit bitwise logic OR	$op1 = op2 \mid \text{Immediate}$	Z	11
XORI op1,op2,Immediate	12 bit bitwise exclusive OR	$op1 = op2 \wedge \text{Immediate}$	Z	11



Mnemonic	Name	Semantics	F	#C
Other Instructions				
CMPA op2,Immediate	comparison	if op2 == Immediate then: Z = 1 else: Z = 0	Z	9
MOVI op1,Immediate	Move immediate	op1[11:0] = Immediate, op1[255:12] = 0		6
HASH_IT	Hash init	Reset hash calculation.		9
GPK op1, Immediate	Get Private Key	op1 = Private key, Key index = immediate		-

Due to not enough space in the 32 bit instruction format, the immediate operand is just 12 bit. Because of that, the logic instructions works only with the 12 LSBs of op2. E.g. 0xFF12 & 0xF0F = 0xFF02.

2.6 M instructions

Mnemonic	Name	Semantics	F	#C
LD op1,Addr	Load	op1[31:0] = Mem[Addr] op1[63:32] = Mem[Addr+0x4] ... op1[255:224] = Mem[Addr+0x1C]		21
ST op2,Addr	Store	Mem[Addr] = op1[31:0] Mem[Addr+0x4] = op1[63:32] = ... Mem[Addr+0x1C] = op1[255:224]		12



2.7 J instructions

Mnemonic	Name	Semantics	F	#C
CALL NewPC	Subroutine call	push(RAR, PC+0x4), PC = NewPC		5
RET	Return from subroutine	PC = pop(RAR)		5
BRZ NewPC	Branch on Zero	if Z == 1 then: PC = NewPC		5
BRNZ NewPC	Branch on not Zero	if Z == 0 then: PC = NewPC		5
BRC NewPC	Branch on Carry	if C == 1 then: PC = NewPC		5
BRNC NewPC	Branch on not Carry	if C == 0 then: PC = NewPC		5
JMP NewPC	Unconditional jump	PC = NewPC		5
END	End of program, stops FW execution and sets STATUS[DONE] .	-		4
NOP	Does nothing.	-		3

3 Flags

3.1 Zero Flag – Z

Zero flag is set to 1, if instruction changing the flag is executed and:

- bits 31:0 of op1 are 0
- $\text{op2}[31:0] - \text{op3}[31:0] = 0$ in case of CMP and CMPI instructions

and cleared otherwise.

Zero flag keeps its value if instruction that does not modify it is executed.

3.2 Carry Flag – C

Carry flag is set to 1, if instruction changing the flag is executed and:

- $\text{op2}[255] = 1$ in case of LSL and ROL instructions
- $\text{op2}[0] = 1$ in case of LSR and ROR instructions

and cleared otherwise.

Carry flag keeps its value if instruction that does not modify it is executed.