SPECT

ISA v0.2

Version: 0.1

Git tag:

Tropic Square

October 18, 2023

1 Glossary

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

2 Instruction set

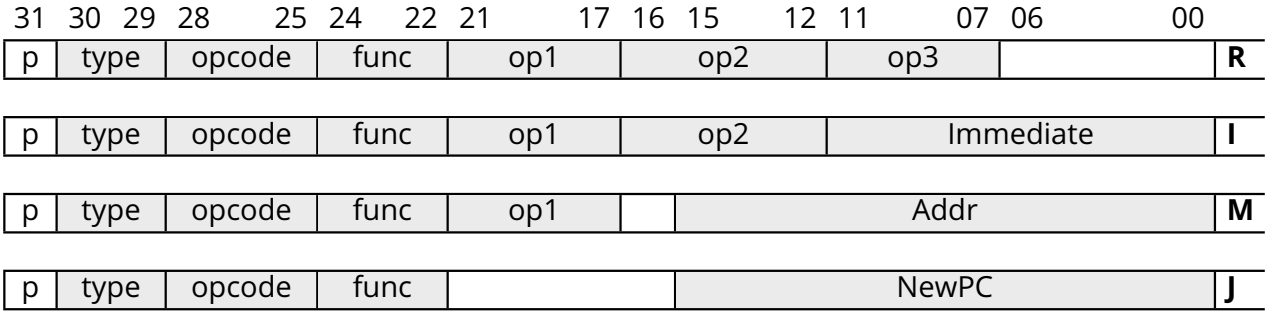
SPECT provides 4 types of instructions:

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

2.1 Operand interpretation

All operands are considered as 256 bits unsigned integers. Arithmetic instructions that work only with 32 bit operands ignores the 224 MSBs of input and clears them in the result. Immediate logic instructions work only with 12 LSBs, ignore the 244 MSBs of input, and pass the 244 MSBs of op2 to the result.

2.2 Instruction Format



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
Arithmetic Instructions (32 bit)				
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
Logic Instructions				
AND op1,op2,op3	Bitwise AND	op1 = op2 & op3	Z	11
OR op1,op2,op3	Bitwise OR	op1 = op2 | op3	Z	11
XOR op1,op2,op3	Bitwise Exclusive OR	op1 = op2 ^ op3	Z	11
NOT op1,op2	Bitwise NOT	op1 = ~op2	Z	10
SBIT op1,op2,op3	Set bit	op1 = op2 $\vee$ (0x1 $\ll$ op3[7:0])		11
CBIT op1,op2,op3	Clear bit	op1 = op2 $\wedge$ $\sim$(0x1 $\ll$ op3[7:0])		11
Shift Instructions				
LSL op1,op2	Logic shift left	op1 = op2[254:0] || 0	C	10
LSR op1,op2	Logic shift right	op1 = 0 || op2[255:1]	C	10
ROL op1,op2	Rotating shift left	op1 = op2[254:0] || op2[255]	C	10
ROR op1,op2	Rotating shift right	op1 = op2[0] || op2[255:1]	C	10
ROL8 op1,op2	Rotating byte shift left	op1 = op2[247:0] || op2[255:248]		10
ROR8 op1,op2	Rotating byte shift right	op1 = op2[7:0] || op2[255:8]		10
ROLIN op1,op2,op3	Rotating byte shift left with shift in from op3	op1 = op2[247:0] || op3[255:248]		11
RORIN op1,op2,op3	Rotating byte shift right with shift in from op3	op1 = op3[7:0] || op2[255:8]		11

Mnemonic	Name	Semantics	F	#C
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
MULP op1,op2,op3	Generic Modular Multiplication	op1 = (op2 * op3) % R31		597
REDP op1,op2,op3	Generic Modular Reduction	op1 = (op2 || op3) % R31		528
Load Instructions				
LDR op1,op2	Load register	op1[31:0] = Mem[op2] op1[63:32] = Mem[op2+0x4] ... op1[255:224] = Mem[op2+0x1C]		–
STR op1,op2	Store register	Mem[op2] = op1[31:0] Mem[op2+0x4] = op1[63:32] ... Mem[op2+0x1C] = op1[255:224]		–
Other Instructions				
MOV op1,op2	Move register	op1 = op2		7
CSWAP op1,op2	Conditional swap – C flag	***if*** **C** == 1 ***then:*** op1 = op2 op2 = op1		11
ZSWAP op1,op2	Conditional swap – Z flag	***if*** **Z** == 1 ***then:*** op1 = op2 op2 = op1		11

Mnemonic	Name	Semantics	F	#C
HASH op1,op2	Hash (SHA512)	Updates SHA core with (op2+3||op2+2||op2+1||op2) op1 = SHA state[255:0] op1+1 = SHA state[511:256]		347
TMAC_IT op2	TMAC initialize	Resets TMAC and underlying KECCAK core mask = (op2+3||op2+2||op2+1||op2) Share A = mask[399:0] Share B = mask[799:0] Guard = [803:800]		94
TMAC_UP op2	TMAC update	Updates TMAC with op2[143:0]		44
TMAC_RD op1	TMAC update	op1 = TMAC result		84
GRV op1	Get Random Value	op1 = Random number		–
SCB op1,op2,op3	Blind scalar	B = *Blind*(op2, op3, R31) op1 = B[255:0] op1+1 = B[511:256]		88

2.5 I instructions

Mnemonic	Name	Semantics	F	#C
Arithmetic Instructions (32 bit)				
ADDI op1,op2,Immediate	32 bit addition	op1 = op2 + Immediate	Z	11
SUBI op1,op2,Immediate	32 bit subtraction	op1 = op2 - Immediate	Z	11
CMPI op2,Immediate	32 bit comparison	op2 - Immediate	Z	9
Logic Instructions (12 bit)				
ANDI op1,op2,Immediate	12 bit bitwise logic AND	op1[11:0] = op2[11:0] & Immediate op1[255:12] = op2[255:12]	Z	11

Mnemonic	Name	Semantics	F	#C
ORI op1,op2,Immediate	12 bit bitwise logic OR	op1[11:0] = op2[11:0] | Immediate op1[255:12] = op2[255:12]	Z	11
XORI op1,op2,Immediate	12 bit bitwise exclusive OR	op1[11:0] = op2[11:0] ^ Immediate op1[255:12] = op2[255:12]	Z	11
KBUS Instructions				
LDK op1,op2,Immediate	Load key	op1 = KBUS_READ[type,slot,offset] where type = Immediate[11:8] slot = op2[7:0] offset = Immediate[4:0] * 8	E	–
STK op1,op2,Immediate	Load key	KBUS_WRITE[key,type,slot,offset] where key = op1 type = Immediate[11:8] slot = op2[7:0] offset = Immediate[4:0] * 8	E	–
KBO op2,Immediate	KBUS OP	KBUS_OP[type,slot,op] where type = Immediate[11:8] slot = op2[7:0] op = Immediate[3:0]	E	–
Other Instructions				
MOVI op1,Immediate	Move immediate	op1[11:0] = Immediate, op1[255:12] = 0		6
HASH_IT	Hash init	Reset hash calculation.		9
TMAC_IS op2, Immediate	TMAC initstring	Initialize TMAC with initstring K = op2, N = Imd[7:0]		78

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]		–
ST op1,Addr	Store	Mem[Addr] = op1[31:0] Mem[Addr+0x4] = op1[63:32] = ... Mem[Addr+0x1C] = op1[255:224]		–

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
BRE NewPC	Branch on Error	***if*** E == 1 ***then:*** PC = NewPC		5
BRNE NewPC	Branch on not Error	***if*** E == 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:

- all 256 bits of op1 are 0
- op2 - op3 = 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 Cary Flag – C

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

- op2[255] = 1 in case of LSL and ROL instructions
- 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.

3.3 Error Flag – E

Error flag is set to 1 in *spect_kbus_error* is set during KBUS request when LDK, STK and KBO instructions are executed.
Error flag keeps its value if instruction that does not modify it is executed.