Operation	Desc	Number of args	Opcode							38 operations		16 registers		64 words				
TRAP	the ALU enters the trap mode, only valid as the rst instruction to be executed, programs cannot jump to this instruction afterwards	0	000001		ALU active Y/N					Opcodes must be 6 bits		reg addrs are 4 bits		mem addrs are 6 bits				
NOP	the ALU spends one CPU clock cycle without producing results	0	100000		Group sel 1		ALU Operation											
JMP	move program pointer to a relative address specied by the contents of a given general purpose register	1	100100		Group sel 2		28 Operations								Instruction format			
JMPZ	move program pointer to a relative address specied by the contents of a given general purpose register if the zero ag is 1	1	100101		Group sel 3						extra	src reg 2	src reg 1	dest reg addr	opcode			
JMPS	move program pointer to a relative address specied by the contents of a given general purpose register if the sign ag is 1	1	100110		Op sel 1					bit locations	19:18	17:14	13:10	9:6	5:0			
JMPZS*	move program pointer to a relative address specied by the contents of a given general purpose register if both the zero and sign ags are 1	1	100111		Op sel 2	Group (first)->	000	001	010	011								
LSTAT	load the contents of the status register into a general purpose register	1	001100			00	NOP	JMP	NOT	SHFTL					Special operations			
XSTAT	xor the value of the status register with the values of the current register and store it only if the trap mode ag is set	1	001101			01	SWAP	JMPZ	AND	SHFTR					MUI and DIV require 4 regs, but that cannot fit in our instructions			
NOT	logical complement of the contents of a register, store it in another register	2	101000			10		JMPS	OR	ROTL			const val	dest reg addr	LDC			
AND		3	101001			11		JMPZS*	XOR	ROTR			19:10	9:6	5:0			
OR		3	101010				100	101	110	111								
XOR		3	101011			00	ADD	INC	LT	MUL*								
SHFTR		3	101101			01	ADDC*	DEC	GT	DIV*								
SHFTL		3	101100			10	SUB	EQ	LET*									
ROTR		3	101111			11	SUBC*	NEQ	GET*									
ROTL		3	101110															
SWAP*	exchange values stored in two different registers; if the registers are the same one, perform a NOP	2	100001				NON ALU											
INC		1	110100				10 operations											
DEC		1	110101			Group (first)->	000	001	010	011								
ADD	add the content of two registers and place the output in another one without taking into account the carry bit	3	110000			00	HALT	MRR	LDD	LSTAT								
ADDC*	add the content of two registers and place the output in another one taking into account the carry bit	3	110001			01	TRAP	LDC	LDI*	XSTAT								
SUB	subtract the content of two registers and place the output in another one without taking into account the carry bit as a borrow	3	110010			10			STD									
SUBC*	subtract the content of two registers and place the output in another one without taking into account the carry bit as a borrow	3	110011			11			STI*									
EQ	compare register A to register B, set zero ag to 1 if equal, 0 otherwise	2	110110				100	101	110	111								
GT	compare register A to register B, set sign ag to 0 if A > B, 1 otherwise	2	111001			00												
LT	compare register A to register B, set sign ag to 1 if A < B, 0 otherwise	2	111000			01												
GET*	compare register A to register B, set sign to 0 and zero ag to 1 if A >= B, invert these otherwise	2	111011			10												
LET*	compare register A to register B, set sign to 1 and zero ag to 1 if A <= B, invert these otherwise	2	111010			11												
MRR	more contents stored in register Ro into register Rd	2	000100															
LDC	load numerical constant C directly into register R	2*	000101															
LDD	load data contained in the origin address contained in Ro to destination register Rd	2	001000															
LDI*	load data to Rd contained in the address contained in the memory address pointed by Roi	2	001001															
STD	store data contained in Ro to the destination in address Rd	2	001010															
STI*	store data contained in Roi to the address contained in the memory address pointed by Rd	2	001011															
MUL*		4!	111100															
DIV*		4!	111101															
HALT		0	000000															
NEQ*		2	110101															