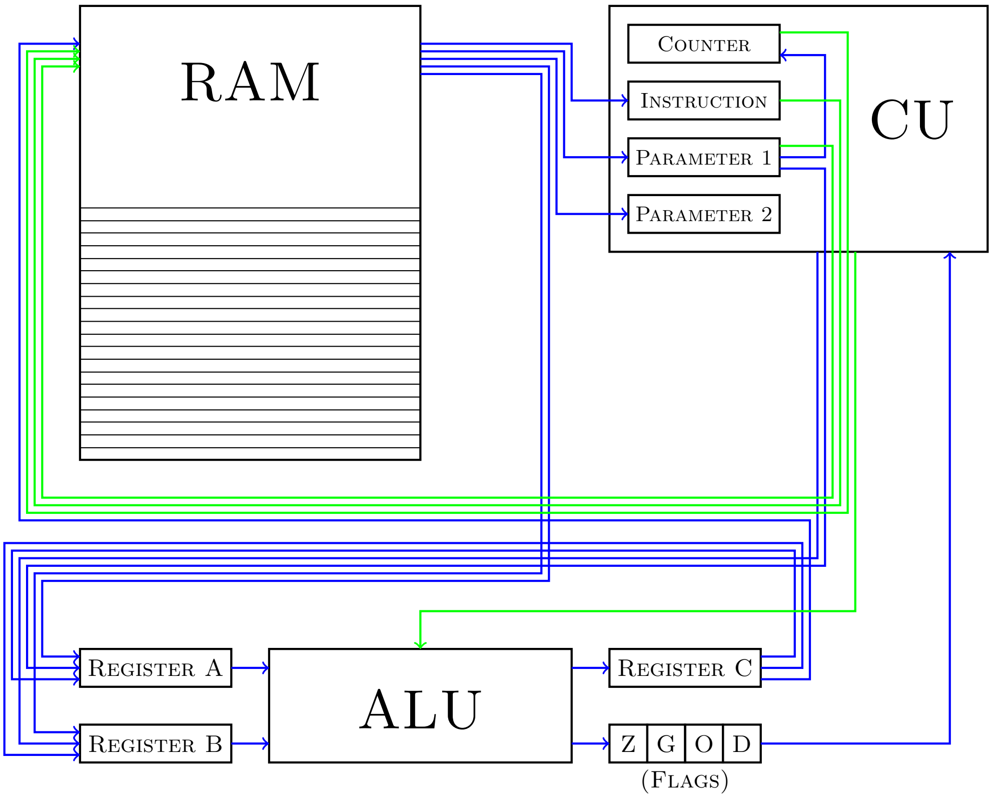

Figure 1: Schematic view of VM. Blue: data, green: control, adresses

Table 1: Overview of all Operations supported by the Virtual Machine

Machine Code	**Assembly Command**	**Number of Paramters**	**Describtion**
0x00	STP	0	Stops the execution
0x01	JMP	1	Unconditional jump
0x02	JGZ	1	Jump if C > 0
0x03	JOF	1	Jump if last Operation caused Overflow
0x04	ADD	0	Computes C = A + B
0x05	SUB	0	Computes C = A - B
0x06	AND	0	Computes C = A & B (logical bitwise and)
0x07	BOR	0	Computes C = A | B (logical bitwise or)
0x08	SHL	0	Shift A one bit to the left, store in C
0x09	SHR	0	Shift A one bit to the right, store in C
0x0A	LDA	1	Load value from adress in A
0x0B	LDB	1	Load value from adress in B
0x0C	LDC	1	Load constant value in C
0x0D	LD0	0	Store 0 in B
0x0E	STR	1	Store value from C in RAM
0x0F	MOV	2	Copy value from first adress to second adress
0x10	NOP	0	No operation
0x11	–	0	Currently not used
0x12	JEZ	1	Jump if C = 0
0x13	JNO	1	Jump if last operation caused no overflow
0x14	MUL	0	Computes C = A · B
0x15	DIV	0	Computes C = A `div` B (integer division)
0x16	–	0	Currently not used
0x17	–	0	Currently not used
0x18	–	0	Currently not used
0x19	–	0	Currently not used
0x1A	RLA	0	Reload value fom C into A
0x1B	RLB	0	Reload value fom C into B
0x1C	LDM	0	Load maximum value into B
0x1D	LD1	0	Load 1 into B
0x1E	–	0	Currently not used
0x1F	–	0	Currently not used

Table 2: Overview of additional Operations supported by Assembler

Assembly Command	**Number of Paramters**	**Describtion**
EAD	2	C = sum of both parameters
ESU	2	C = first parameter - second parameter
EMU	2	C = product of both parameters
EDI	2	C = first parameter `div` second parameter (integer division)
STC	2	store second parameter at adress specified by first parameter in RAM

Example program `Prim.txt`

```
// a test program that checks if the value in test is prim
// writes 1 to res if test is prim
// writes 0 to res otherwise
VAR test
VAR max
VAR counter
VAR res
BEGIN
STC 47 test
STC 2 counter
LDA test
SHR
// max = test/2, maximum number that needs to be checked
STR max
LABEL start
// check if test/counter has remainder:
LDA test
LDB counter
DIV
RLA
MUL
RLA
LDB test
SUB
// if 0, no remainder: test is not prim
JEZ notprim
// otherwise increase counter
LDA counter
LD1
ADD
STR counter
RLB
// check if max is reached:
LDA max
SUB
// counter < max: goto start
JGZ start
// otherwise: test is prim
STC 1 res
JMP end
LABEL notprim
STC 0 res
LABEL end
STP
```