Toy virtual machine

Write a virtual machine simulator that with the following requirements met.

The machine has 4 registers with names R1 -- R4. Each register is 8 bit integer. The machine has 256 byte RAM. The machine takes a code from a file defined as a command line argument. The code is defined as a text, e.g.:

```
MOV R1, 61
MOV R2, 22
ADD R1, R2
```

When the simulator reached last instruction it prints all the registers it has (optionally for hidden ones) and exits.

The machine must support the following instruction groups. The first argument is *always* a destination.

Syntax description

Labels

Labels are optional and must either start with a letter or a dot (.) and end with a colon. Label could be put alone in a line or with an instruction. Comments are optional and start with `;`.

```
label: instruction operands ; Comment
```

Operands format

Operands could be defined in three different formats:

```
reg      -> Register: R1, R2, R3, R4
address  -> Memory address: [15] ; 15th byte
constant -> Either a number or named constant: 42, constant
```

MOV - Copy a value

Copies a value from *src* to *dest*.

```
MOV reg, reg
MOV reg, address
MOV reg, constant
```

```
MOV address, reg
MOV address, constant
```

DB - Named constant

Defines a named constant.

`DB constant`

Math operations

Addition and Subtraction

Adds two numbers together or subtract one number form another.

```
ADD reg, reg
ADD reg, address
ADD reg, constant
SUB reg, reg
SUB reg, address
SUB reg, constant
```

Multiplication and division

Multiplies or divides the *R1* register with the given value.

```
MUL reg
MUL address
MUL constant
DIV reg
DIV address
DIV constant
```

Shift instructions

Shift value to the left or to the right on the defined number of bits. SHL shifts to the left, SHR shifts to the right.

```
SHL reg, reg
SHL reg, address
SHL reg, constant
SHR reg, reg
SHR reg, address
SHR reg, constant
```

Logical instructions

The following logical instructions are supported: AND, OR, XOR, NOT.

```
AND reg, reg
AND reg, address
AND reg, constant
OR reg, reg
OR reg, address
OR reg, constant
XOR reg, reg
XOR reg, address
XOR reg, constant
NOT reg
```

CMP - Compare

Compares two values. Use this instruction before a conditional jump.

```
CMP reg, reg
CMP reg, address
CMP reg, constant
```

Jumps

JMP - Unconditional jump

Let the instruction pointer do a unconditional jump to the defined address.

```
JMP address
```

Conditional jumps

Just if last compare instruction has corresponding result. JE is jump if equal, JL is jump if less, JG is jump if greater.

```
JE address
JL address
JG address
```

Subroutines

CALL - Function call

Call is used to jump into subroutine with the possibility to return back to the calling address+1.

```
CALL address
```

RET - Exit a subroutine

Exits a subroutine and jump back to the calling address+1.

```
RET
```

Stack instructions

PUSH - Push to stack

Pushes a value to the stack.

```
PUSH reg
PUSH address
PUSH constant
```

POP - Pop from stack

Pops a value from the stack to the register.

```
POP reg
```

Other instructions

HLT - Stops the machine

Stops the execution and prints all the registers defined (optionally hidden).

```
HLT
```