1. SASM Instructions

File: `sasm_instructions.h`
Author: Soham Metha
Date: January 2025

1.1. Table of Contents

1.2. Opcode Enumeration

The `Opcode` enum defines all supported instructions in SASM.

Opcode	Description
NOP	No operation
HLT	Halt
ADD	Addition
SUB	Subtraction
MUL	Multiplication
DIV	Division
EQL	Equality
POP	Pop
PSH	Push
DUP	Duplicate
JMP	Jump
JNZ	Jump if not zero

Opcode	Description
JIP	Jump if positive

1.3. Data Structures

1.3.1. Instruction Structure

Represents a single SASM instruction, consisting of an opcode and an operand.

```
typedef struct {
    Opcode type;  /**< The opcode of the instruction */
    Word operand; /**< The operand of the instruction */
} Instruction;
```

1.3.2. Program Structure

Represents a complete SASM program, including instructions and metadata.

```
typedef struct {
    Instruction instructions[PROGRAM_CAPACITY]; /**< Array of
instructions */
    Word instruction_count;                     /**< Number of
instructions */
    Word instruction_size;                      /**< Size of each
instruction in bytes */
} Program;
```

1.4. Function Declarations

1.4.1. Stack Operations

1. **Push to Stack**

   ```
   Error __psh(Registers* r, Memory* mem, const Word* operand);
   ```

   Pushes a value onto the stack.

2. **Equality Operation**

   ```
   Error __eql(Registers* r, Memory* mem);
   ```

Compares the top two values of the stack for equality.

3. **Pop from Stack**

```
Error __pop(Registers* r, Memory* mem);
```

Pops a value from the stack.

4. **Duplicate Value**

```
Error __dup(Registers* r, Memory* mem, const Word* operand);
```

Duplicates a value on the stack based on the operand.

1.4.2. Instruction Execution

- **Execute an Instruction**

```
Error executeInst(const Program* prog, Memory* mem, CPU* cpu);
```

Executes a single instruction in the virtual machine.

1.4.3. Opcode Conversion

1. **Convert Opcode to String**

```
String opcodeAsStr(const Opcode* type);
```

Converts an opcode to its string representation.

2. **Convert String to Opcode**

```
Opcode strAsOpcode(const String* s);
```

Converts a string to its corresponding opcode.