SoftArt Intermediate Shader Language Virtual Machine指令集、指令生成工具、执行机设计文档

该文档将主要包含以下内容：

* SAISLVM指令格式
* SAISLVM指令名与指令微码常数命名规则
* SAISLVM项目组织
* SAISLVM指令集的C++实现与扩展
* SAISLVM指令执行引擎实现与扩展
* SAISLVM弱类型指令生成器实现与扩展
* SAISLVM强类型指令生成器实现与扩展

# SAISLVM指令格式

SoftArt Intermediate Shader Language Virtual Machine（SISLVM，以下简称VM）的助记符指令格式采用三元组的形式。

|  |  |  |
| --- | --- | --- |
| 指令名 | 操作数1 | 操作数2 |

VM的二进制指令格式类似于助记符指令格式。

|  |  |  |
| --- | --- | --- |
| 指令微码 | 操作数1 | 操作数2 |

VM在32位和64位条件下，将分别编译成32位和64位两个不同的版本。

32位的VM和64位的VM产生的字节码文件互不通用，同时32位下不支持64位指令。

下表为VM的指令格式。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 32位 VM | | | 64位 VM | | |
| 指令微码 | 操作数1 | 操作数2 | 指令微码 | 操作数1 | 操作数2 |
| Size | | 4 B | 4 B | 4 B | 4 B | 8 B | 8 B |
| Aligned Size | | 4 B | 4 B | 4 B | 8 B | 4 B | 4 B |
| Raw Type | | Signed Integer | Signed Integer | Signed Integer | Signed Integer | Signed Integer | Singed Integer |
| Offset Type | | N/A | Signed Integer | Signed Integer | N/A | Signed Integer | Signed Integer |
| Offset Size | | N/A | 4 B | 4 B | N/A | 8 B | 8 B |
| Pointer Size | | N/A | 4 B | 4 B | N/A | 8 B | 8 B |
| Supported Opreand Types | Int8 | N/A | ● | ● | N/A | ● | ● |
| UInt8 | N/A | ● | ● | N/A | ● | ● |
| Int16 | N/A | ● | ● | N/A | ● | ● |
| UInt16 | N/A | ● | ● | N/A | ● | ● |
| Int32 | N/A | ● | ● | N/A | ● | ● |
| UInt32 | N/A | ● | ● | N/A | ● | ● |
| Int64 | N/A | — | — | N/A | ● | ● |
| UInt64 | N/A | — | — | N/A | ● | ● |
| Float | N/A | ● | ● | N/A | ● | ● |
| Double | N/A | — | — | N/A | — | — |
| Offset | N/A | ● | ● | N/A | ● | ● |
| Pointer | N/A | ● | ● | N/A | ● | ● |

# SAISLVM SAISLVM指令名与指令微码常数命名规则

指令根据功能直接命名。指令名称不包含潜在的类型、参数、上下文等信息。名称较长的指令名，可使用缩写，如**Shift Left**简记为*shl*。

在SASL VM中，有效的指令名（助记符）及其语义如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| **指令名** | **描述** | **指令名** | **描述** |
| ADD | Add | Load | Copy |
|  |  |  |  |

相同的指令名会有多种不同参数类型的重载。其中每种重载方式拥有唯一的指令微码（Opcode）。在VM中，将会提供这些微码的具名常数。VM不保证指令更新时的向下兼容性，即不同版本的VM相同指令的Opcode不保证是一致的。Opcode的具名常数将有以下部分构成：

指令名\_参数1存储类型\_参数1值类型\_参数2存储类型\_参数2值类型

参数中可能使用的存储类型包括：

|  |  |
| --- | --- |
| **存储类型简称** | **存储类型全称** |
| a | Physical Address |
| s | Stack Base Pointer Based Offset |
| gr | General Register |
| fr | Float Register |
| dr | Double Register |
| \_ | Unused |
| c | Constant |

参数值类型包括

|  |  |
| --- | --- |
| **值类型简称** | **值类型全称** |
| *i8* | 8-bit Signed Integer |
| *i16* | 16-bit Signed Integer |
| *i32* | 32-bit Signed Integer |
| *i64* | 64-bit Signed Integer |
| *ui8* | 8-bit Unsigned Integer |
| *ui16* | 16-bit Unsigned Integer |
| *ui32* | 32-bit Unsigned Integer |
| *ui64* | 64-bit Unsigned Integer |
| *f* | single precision float(32-bit on x86) |
| *d* | double precision float(64-bit on x86) |
| *raw* | 32-bit signed integer(on 32-bit VM) or  64-bit signed Integer(on 64-bit VM) |
| *uraw* | 32-bit unsigned integer(on 32-bit VM) or  64-bit unsigned Integer(on 64-bit VM) |
| *rid* | Register ID. Data Type is same to*raw* |
| *\_* | unused |

例如*load\_gr\_i32\_fr\_f*就是一个合法的指令微码常数名。

# SAISLVM指令集的C++实现与扩展

SAISLVM使用C++实现。工程按照目录结构划分如下

**SASL Project**

* *include*
  + code\_generator：SAISLVM强类型/弱类型指令产生工具
  + parser
  + parser\_tree
  + syntax\_tree
  + vm：SAISLVM虚拟机
    - op\_code：指令集、默认的指令处理、指令转发、指令产生宏
* *source*：结构同include
* *doc*：提供了SASL的相关文档
* *enums*：提供了SASL所需要的类型安全的枚举值。所有代码均为代码生成器生成并不可修改。

**指令集定义**

指令集定义为多个指令的集合，其中每个指令定义为用于描述指令相关信息的元组。指令元组按照以下格式组织内容：

指令信息元组 ::= ‘(‘ 指令名, 参数1存储类型, 参数1值类型, 参数2存储类型, 参数2值类型 ‘)’

指令集按照以下方式定义：

指令集 ::= ( ‘(‘指令信息元组‘)’ )\*

指令集定义在include/vm/op\_code/instruction\_list.h中，通过在宏***SASL\_VM\_INSTRUCTIONS***中添加新的指令信息元组项，即可扩充新的指令。

**指令微码常数名的存储类型修饰**

指令微码常数名操作数值类型修饰宏

指令集定义宏

SASL\_VM\_INSTRUCTIONS

指令集

指令微码常数名宏

指令名宏

带命名空间修饰的指令微码常数宏

带命名空间的存储类型宏

带命名空间的操作数值类型宏