

CoreGen Intermediate Representation Language Specification

Tactical Computing Laboratories, LLC

Date: August 27, 2019
Revision: 1.4.4
Authors: Tactical Computing Labs
contact@tactcomplabs.com

Contents

1 Overview	5
1.1 Required Packages and Manipulating IR	10
2 CoreGen IR Nodes	11
2.1 Node Names and Linkage	11
2.2 User Defined RTL	12
2.3 Node Override	14
2.4 Node Notes	15
2.5 Projects	16
2.6 Registers	17
2.7 Register Classes	19
2.8 Instruction Sets	20
2.9 Instruction Formats	21
2.10 Instructions	23
2.11 Pseudo Instructions	25
2.12 Caches	26
2.13 Scratchpad Memories	27
2.14 Virtual to Physical Translation Units	28
2.15 Memory Controllers	29
2.16 Communication Channels	30
2.17 Cores	31
2.18 SoCs	32
2.19 Extensions	33
2.20 Plugins	35
3 Appendix A: Sample IR	38

List of Figures

1	CoreGen IR Node Hierarchy	9
---	-------------------------------------	---

Listings

1	Example IR YAML Indentation	6
2	Example IR Node Linkage	6
3	Node Naming and Linkage	12
4	Sample Inline RTL	12
5	Sample External RTL File	13
6	Sample Node Override	14
7	Sample Node Notes	15
8	Project Node Definition	16
9	Register Node Definition	18
10	Register Class Node Definition	19
11	Instruction Set Node Definition	20
12	Instruction Format Node Definition	21
13	Instruction Node Definition	24
14	Pseudo Instruction Node Definition	25
15	Cache Node Definition	26
16	Scratchpad Node Definition	27
17	VTP Node Definition	28
18	Memory Controller Node Definition	29
19	Communication Node Definition	30
20	Core Node Definition	31
21	SoC Node Definition	32
22	Extension Node Definition	34
23	Plugin Node Definition	36
24	Sample IR File	38

List of Tables

2	CoreGen IR Scalar Types	11
3	CoreGen IR Valid and Invalid Node Names	11
4	CoreGen Project Types	16
5	CoreGen Register Node Parameters	17
6	CoreGen Instruction Format Node Parameters	21
7	CoreGen Instruction Format Node Parameters	30
8	CoreGen Plugin Node Parameters	35
9	CoreGen Plugin Node Feature Types	36

Revision History

Revision	Date	Author(s)	Description
1.0	09.12.2018	JLeidel	Initial public release
1.1	11.21.2018	JLeidel	Adding ThreadUnits to Core nodes and TUSReg register attributes
1.2	11.21.2018	JLeidel	Adding subregister pseduoregister sub-node structure for setting up register aliases to specific bit regions of parent registers
1.3	12.20.2018	JLeidel	Adding clarifying text to indicate the permissible characters utilized in IR node names
1.4	01.21.2019	JLeidel	Adding RTL Type fields for overloaded RTL models; Adding Syntax mnemonic fields for instructions
1.4.1	04.24.2019	JLeidel	Adding PC attribute to register node attributes
1.4.2	07.03.2019	JLeidel	Adding Override keyword to all candidate hardware nodes: Section 2.3
1.4.3	08.21.2019	JLeidel	Adding syntax mnemonics to pseudo instructions
1.4.4	08.27.2019	JLeidel	Adding notes syntax to each candidate IR node

1 Overview

The CoreGen Intermediate Representation (IR) specification is utilized as the core mechanism by which to translate a hardware design to various levels of hardware design language and simulation infrastructure. The CoreGen IR has the following goals:

- **Support Rapid Design Workflows:** Traditional design techniques and tools require long design and prototyping cycles, often with expensive external tools. The CoreGen IR is design to provide rapid design and prototyping workflows such that hardware and software artifacts can be generated in minutes rather than days.
- **Preserve Hardware Dependencies:** The CoreGen IR is designed to preserve the dependency between individual hardware modules and their constituent submodules. These dependencies are preserved in a manner similar to traditional compiler technology, e.g. a *directed acyclic graph* (DAG). These dependencies are subsequently utilized to analyze certain inherent properties of the DAG in order to verify, optimized and document the design.
- **Support Modular Design Principles/Reuse:** One of the principle issues that plague previous design workflows and paradigms is the lack of potential reuse. The CoreGen IR provides modular reuse of individual design elements as well as the ability to use design templates to rapidly build similar, but specialized design elements. Further, all individual hardware modules can be overridden using external design elements in order to further construct specialized designs.
- **Support High-Level Design Verification:** One of the primary goals of the CoreGen infrastructure is the ability to perform high performance, low latency design verification without the need to perform low-level synthesis and simulation. This capability is provided by the inherent ability to express complex designs within the CoreGen IR.
- **Support Multiple Artifact Generation:** The final goal of the CoreGen IR is to drive secondary and, potentially, external tooling. These tools provide the ability to generate multiple levels of hardware design language for the complementary architecture as well as the potential to generate software artifacts such as compiler tool chains from the design description.

The CoreGen IR is constructed using the standard *YAML* [1] format. This is a human readable text structure that preserves both descriptions of individual hardware modules and references between adjacent modules (the DAG from above). *YAML* is formatted using blocks of *Collections*. Collections describe an individual type of node (hardware module) in the CoreGen IR. For example, the notion of a register file is encapsulated in a single collection. Each collection may contain a set of *sequences*. These sequences are individual instances of the appropriate module type are denoted with a name followed by a colon (NAME:). For example, your design may include multiple different register files, each represented as a sequence. Each sequence will include some number of elements, each denoted with a name followed by a dash (NAME-). These sequence instances may also include a hierarchical set of sequences with scalar values. The named elements and their associated scalar values are marked using the name and colon designator (NAME:). These sub-sequences and associated scalars represent the various design parameters for the target instance of the target node type. In our example, a register file may contain some number of registers. Each register has a name, index value, bit width and other such parameters. Each specific node type includes unique, recognizable scalar parameters (detailed in the sections below). The scalar values assigned to each parameter may include text, boolean values (`true`, `false`), integers and floating point values.

The *YAML* formatting paradigm uses indentation (spaces, *NOT* tabs) in order to define hierarchies and inheritance. For example, in our register node description, you'll find that there is a top-level `Register` designator with multiple sequences of registers, each marked with a top-level `RegName` to denote an individual register. We see an example of this in Listing 1.

```

1 Registers:
2   - RegName: TEST46.0.reg
3     Width: 64
4     Index: 0
5     PseudoName: pseudoreg.0
6     IsFixedValue: false
7     IsSIMD: false
8     RWReg: true
9     ROReg: false
10    CSRReg: true
11    AMSReg: false
12   - RegName: TEST46.1.reg
13     Width: 64
14     Index: 1
15     PseudoName: pseudoreg.1
16     IsFixedValue: false
17     IsSIMD: false
18     RWReg: true
19     ROReg: false
20     CSRReg: true
21     AMSReg: false

```

Listing 1: Example IR YAML Indentation

In addition to the basic ability to describe a design in the CoreGen IR YAML, the IR format also has the ability to describe the inherent connectivity and dependence between hardware modules. The naming conventions utilized by each individual instance of a node can be utilized to link modules together in order to preserve module dependence or physical hardware connectivity. For example, a register class utilized to describe inputs for a given instruction may have dependencies on individual registers. Likewise, an instruction format may require register classes as inputs. An example of describing this hierarchy is show in Listing 2.

```

1 Registers:
2   - RegName: TEST46.0.reg
3     Width: 64
4     Index: 0
5     PseudoName: pseudoreg.0
6     IsFixedValue: false
7     IsSIMD: false
8     RWReg: true
9     ROReg: false
10    CSRReg: true
11    AMSReg: false
12   - RegName: TEST46.1.reg
13     Width: 64
14     Index: 1
15     PseudoName: pseudoreg.1
16     IsFixedValue: false
17     IsSIMD: false
18     RWReg: true
19     ROReg: false
20     CSRReg: true
21     AMSReg: false

```

```

22 RegClasses:
23   - RegisterClassName: TEST46.regclass
24   Registers:
25     - TEST46.0.reg
26     - TEST46.1.reg
27 InstFormats:
28   - InstFormatName: TEST46.if
29   ISA: TEST46.isa
30   FormatWidth: 32
31   Fields:
32     - FieldName: opcode
33       FieldType: CGInstCode
34       FieldWidth: 8
35       StartBit: 0
36       EndBit: 7
37       MandatoryField: true
38     - FieldName: RB
39       FieldType: CGInstReg
40       FieldWidth: 8
41       StartBit: 8
42       EndBit: 15
43       MandatoryField: false
44       RegClass: TEST46.regclass
45     - FieldName: RA
46       FieldType: CGInstReg
47       FieldWidth: 8
48       StartBit: 16
49       EndBit: 23
50       MandatoryField: false
51       RegClass: TEST46.regclass
52     - FieldName: RT
53       FieldType: CGInstReg
54       FieldWidth: 8
55       StartBit: 24
56       EndBit: 31
57       MandatoryField: false
58       RegClass: TEST46.csrregclass

```

Listing 2: Example IR Node Linkage

The CoreGen IR unique IR node types to represent each individual style of hardware module. We briefly list the distinct node types as follows (each of with its own set of unique parameters):

- **SoC:** A *System on Chip*, or SoC node encapsulates a full SoC design. It is effectively a container for larger designs.
- **Core:** A core node contains the necessary arithmetic and register logic to implement a single core.
- **Instruction Format:** An instruction format node contains the necessary logic to define an instruction encoding.
- **Instruction:** An instruction is a representation of a single operation to take place within a core. This may include arithmetic, memory or other styles of operations.
- **Pseudo Instruction:** Pseudo instructions are templated versions of individual instructions, each

with a unique name. These are generally utilized to make an instruction set more expressive without expanding the encoding space.

- **Register Class:** Register class nodes are containers for some number of individual registers that are encapsulated into a hardware register file.
- **Register:** Register nodes are utilized to represent individual hardware registers
- **Instruction Set Architecture:** An instruction set architecture node is utilized to encapsulate multiple instructions and their constituent encodings.
- **Cache:** Cache nodes are utilized to represent an individual layer in a caching hierarchy (for example, an instruction cache or L1 data cache).
- **Encoding:** Encoding nodes are utilized to contain the encodings for individual fields of various structures (such as instruction formats)
- **Communication Link:** Communication nodes are utilized to represent physical links between individual modules.
- **Scratchpad:** Scratchpad nodes are utilized to represent special cases of embedded memories that are contained within an SoC.
- **Memory Controller:** Memory controller nodes are utilized to represent controller modules that connect to external memories.
- **Virtual to Physical Translation Unit:** Virtual to physical translation nodes are utilized to represent the virtual to physical translation mechanisms present within an SoC.
- **Extension:** Extensions are special node types in CoreGen. Extensions have the ability to extend existing modules or module hierarchies that are considered to be self contained. These nodes are utilized to extend existing designs in a modular fashion within breaking the internal dependencies present in already proven designs.
- **Plugin:** Plugin nodes are special node types in CoreGen. Plugins can be utilized to represent modules that do not follow the predefined node types already defined in CoreGen. Plugins can have arbitrary set of parameters that are defined by the plugin architecture. These parameters can be utilized to steer custom code generation mechanisms for HDL and other artifacts. Plugins can also be utilized to override any existing CoreGen node in a templated fashion.

Each of the aforementioned node types follows a predefined hierarchy and dependence profile. We represent these dependencies in Figure 1. The dependencies are maintained via the standard CoreGen IR formatting. As a result, dependencies are naturally expressed in the IR.

The two exceptions to this rule are *Extensions* and *Plugins*. These can effectively fall anywhere within the IR hierarchy. Further, both extensions and plugins can contain dependent nodes within themselves in order to express additional information. We will further see how these function in Sections 2.19 and 2.20.

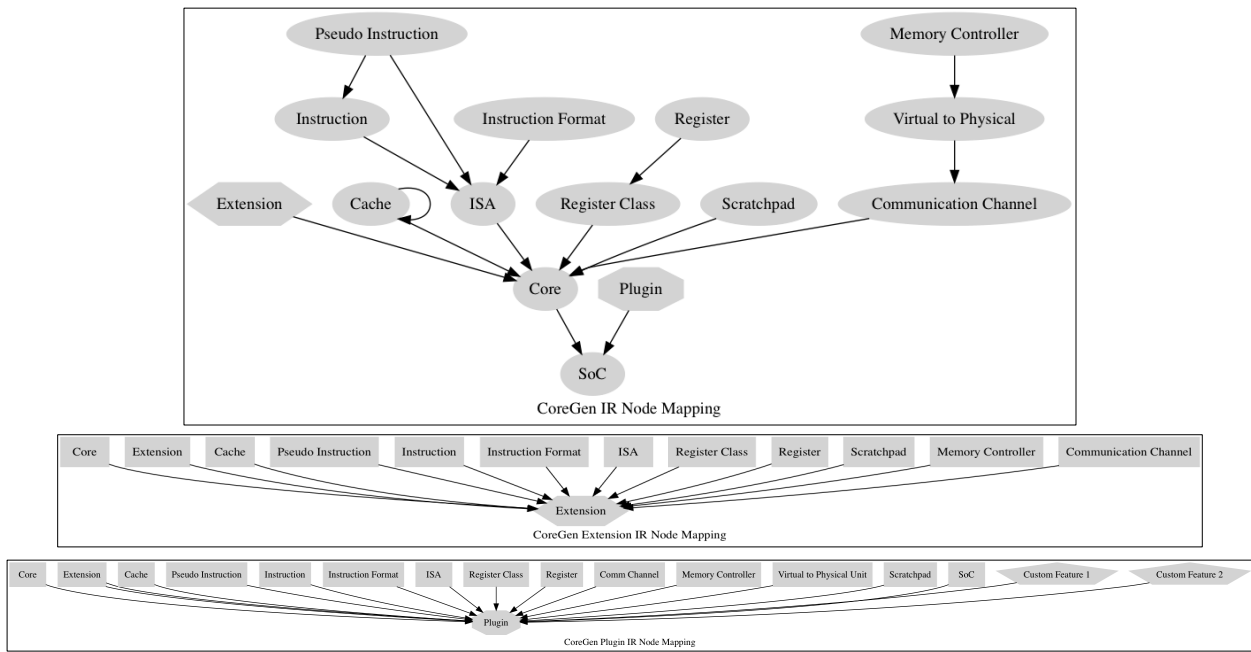


Figure 1: CoreGen IR Node Hierarchy

1.1 Required Packages and Manipulating IR

Given the textual nature of the CoreGen intermediate representation, there are no external packages required to read and/or write the IR. However, we *highly* recommend that you utilize the integrated CoreGen tooling in order to maintain the internal dependencies and verify the syntactical correctness of the IR. Further, parsing the YAML using non-CoreGen tools may result in unresolved internal dependency links due to incorrect parsing order. Regardless of the format of the IR text, the CoreGen tools will parse the file in the correct nodal order.

2 CoreGen IR Nodes

In this section, we outline how each individual node type is constructed, how they link to other nodes and the syntactical definitions utilized therein. Each section describes the generic format of the node. The sequences and sequence elements will include scalar values in one of the following types (Table 2):

Table 2: CoreGen IR Scalar Types

Type	Description
String	Strings are utilized to represent names and references to names. In some instances, Strings are used to represent special object types.
Bool	Bool values are utilized to represent pure booleans. They can be either <code>true</code> or <code>false</code> .
Integer	Integer values are utilized to represent indices and basic parameter values. Where appropriate, we indicate the perceived types.
Special	Special objects a textual keywords used to trigger specific architectural features. These are represented as: $\{\text{Value1}, \text{Value2}, \dots\}$.

2.1 Node Names and Linkage

Each IR node in the CoreGen IR format has a unique name that represents a unique hardware object. Names must be unique across all IR nodes. We require unique names as the respective node names are utilized to link nodes in a dependent manner across the design. For example, a `core` node may have a dependent `cache` node. We provide an example node dependency using these names in Listing 3. Notice how we define an instruction node (`TEST54.inst0`) and a pseudo instruction node (`TEST54.pinst0`) that depends upon the `core` instruction by name.

Given this use of textual names in the IR, we must define a valid naming convention that is preserved in the IR. The valid node name rules are listed as follows:

- Names must be non null (*length greater than 0*)
- Names must begin with an alpha character ($[a-z, A-Z]$)
- Names must contain only alphanumeric characters and periods ($[a-z, A-Z, 0-9, .]$)
- Names are case sensitive, but can include any combination of upper and lower case characters

Example valid and invalid node names are listed in Table 3.

Table 3: CoreGen IR Valid and Invalid Node Names

Valid	Invalid
<code>nodename</code> , <code>nodeName</code> , <code>nodeName0</code> , <code>Node.Name.9</code>	<code>9Name</code> , <code>Node&Name</code> , <code>%nodeName</code>

```

1 Insts:
2   - Inst: TEST54.inst0
3     ISA: TEST54.isa
4     InstFormat: TEST54.if
5     Encodings:
6       - EncodingField: opcode
7         EncodingWidth: 8
8         EncodingValue: 1
9 PseudoInsts:
10  - PseudoInst: TEST54.pinst0
11    ISA: TEST54.isa
12    Inst: TEST54.inst0
13    Encodings:
14      - EncodingField: RA
15        EncodingWidth: 8
16        EncodingValue: 0

```

Listing 3: Node Naming and Linkage

2.2 User Defined RTL

For each of the forthcoming node types, users have the ability to override the default CoreGen code generation facilities in order to include custom RTL. Currently, The CoreGen IR specification permits users to utilize this feature in two ways. First, users have the ability to define inline RTL. This RTL can currently be written in either Chisel (Chisel 3) or Verilog. Inline RTL must follow the naming conventions and port definitions supported by the CoreGen code generation facilities (NEED REFERENCE). An example of defining inline RTL for a single instruction in Chisel3 is shown in Listing 4. Notice how the inline RTL source code is quoted. For each inline RTL sample provided with the CoreGen node, you must also specify the RTLType. Types can be one of Chisel, Verilog and Unknown.

```

1 Insts:
2   - Inst: TEST69.inst0
3     ISA: TEST69.isa
4     InstFormat: TEST69.if
5     Encodings:
6       - EncodingField: opcode
7         EncodingWidth: 8
8         EncodingValue: 1
9     RTL: "when( RA === 0x05) {
10         RT = RA << RB
11       } .otherwise {
12         RT = RA >> RB
13       }"
14     RTLType: Chisel

```

Listing 4: Sample Inline RTL

The second path by which users may override a default node using RTL is by utilizing an external RTL file. By utilizing the RTLFile parameter with any node, the user may specify an external RTL file that is managed outside of CoreGen and its constituent tools. Take note that the external RTL file is relative to the project path utilized by the CoreGen tools. An example of doing so is shown in Listing 5.

```
1 Insts:  
2   - Inst: TEST69.inst0  
3     ISA: TEST69.isa  
4     InstFormat: TEST69.if  
5     Encodings:  
6       - EncodingField: opcode  
7         EncodingWidth: 8  
8         EncodingValue: 1  
9     RTLFile: Sample.v  
10    RTLType: Verilog
```

Listing 5: Sample External RTL File

2.3 Node Override

In addition to providing the ability to override each node with user-defined RTL, the CoreGen IR infrastructure also provides the ability to override an entire node's construction with user-defined logic. This mechanism is analogous to utilizing templates in modern object-oriented programming models. Users have the ability to utilize *plugins* to override the default behavior and/or define non-standard features for a target IR node. This feature can also be utilized to integrate pre-existing hardware IP into new designs. This functionality can be performed on all nodes except plugins, extensions, encoding and pseudo instructions.

An example of utilizing this feature is provided in Listing 6. The first step in this process is to define a plugin node that will be utilized to override the default functionality of a standard node. In this case, we utilize a pre-existing `CacheLayerPlugin` and define a `CachePlugin` node. This theoretical plugin node instantiates an N -level cache with pre-existing IP where the number of cache levels is defined by a custom feature (`Levels`). With this plugin node defined, we can now define a base cache node defined as the `L0.Cache`. Within this cache node, we use the `Override` keyword and attach the plugin node definition that we previously defined (`CachePlugin`). Note that you **must** utilize the plugin node name (`Plugin`), not the actual name of the plugin implementation (`PluginName`). The CoreGen IR supports multiple instantiations of the same plugin implementation under different names. This permits users to re-use existing IP and define unique features for each instance of the plugin. For more information regarding the definition and construction of the actual `Plugin` nodes, see Section 2.20.

```

1 Plugins:
2   - Plugin: CachePlugin
3     PluginName: CacheLayerPlugin
4     MajorVersion: 1
5     MinorVersion: 0
6     PatchVersion: 0
7     Features:
8       - FeatureName: Levels
9         FeatureType: unsigned
10        FeatureValue: 3
11 Caches:
12   - Cache: L0.Cache
13     Sets: 1024
14     Ways: 8
15     Override: CachePlugin

```

Listing 6: Sample Node Override

2.4 Node Notes

Each node has an additional attribute that permits users to add specific *notes* or comments for future reference. These `Notes` fields do not affect the overall functionality, stability or connectivity of the constituent design. Rather, these attributes may be utilized, queried or processed by downstream tools. For example, the *CoreGen SpecDoc* pass utilizes these notes to output specific information for constituent design elements. An example of doing so is shown in Listing 7.

```
1 Insts:  
2   - Inst: TEST69.inst0  
3     ISA: TEST69.isa  
4     InstFormat: TEST69.if  
5     Encodings:  
6       - EncodingField: opcode  
7         EncodingWidth: 8  
8         EncodingValue: 1  
9     Notes: This instruction performs carry-less addition.
```

Listing 7: Sample Node Notes

2.5 Projects

CoreGen IR files may provide an optional project definition block. This block of IR does not contain any dependencies with other IR nodes. Rather, it is designed to provide tools with additional information regarding the location and target function of the IR contained therein. Project nodes are utilized to define project names, paths, project types and Chisel output versions. Currently, we support four disparate project types as show in Table 4.

Table 4: CoreGen Project Types

Parameter	Description
soc	System on chip projects are full designs with deep hierarchies of nodes
module	Module projects are designed to be very compact hierarchies of nodes potentially with custom RTL
extension	Extension projects encapsulate a single block of extension IR nodes to be utilized in other projects
unknown	All other projects fall into the <i>unknown</i> category

We see an example block of project node IR in Listing 8.

```

1 ProjectInfo:
2   - ProjectName: String
3   ProjectRoot: String
4   ProjectType: Special
5   ChiselMajorVersion: Integer
6   ChiselMinorVersion: Integer

```

Listing 8: Project Node Definition

2.6 Registers

Register nodes define a single instance of a register in hardware. Registers may also have named aliases, referred to as a `PseudoName`. Further, subregisters (`SubRegs`) can be defined whereby a unique pseudoname is utilized to access a specific portion of the parent register. For example, if we define a parent register `FOO` with a width of 64 bits, we may also define a `SubReg FOO.BAR` to alias bits `[31-0]` (the lower half of the register).

Additional parameters are described in Table 5. We provide a sample register IR block in Listing 9.

Table 5: CoreGen Register Node Parameters

Parameter	Description
<code>Width</code>	Defines the width (in bits) of the register. (Int)
<code>Index</code>	Defines the index value of the register. (Int)
<code>PseudoName</code>	[Optional] Defines a register pseudo name (String)
<code>IsFixedValue</code>	Sets a parameter indicating that the register has a read-only, fixed value. (Bool).
<code>IsSIMD</code>	Sets a parameter indicating that the register is a SIMD register. The register width must encapsulate the entire SIMD block. (Bool)
<code>RWReg</code>	Sets a parameter indicating that the register is a read-write register (Bool)
<code>ROReg</code>	Sets a parameter indicating that the register is a read-write register. Cannot be <code>true</code> if <code>RWReg</code> is <code>true</code> . (Bool)
<code>CSRReg</code>	Sets a parameter indicating that the register is a control status register. This can be read-only or read-write. (Bool)
<code>AMSReg</code>	Sets a parameter indicating that the register is an arithmetic machine state register. The hardware code generator will integrate these into the pipeline. (Bool)
<code>TUSReg</code>	Sets a parameter indicating that the register is shared across all thread units within the core. Either <code>Shared</code> or <code>TUSReg</code> can be set, but not both. (Bool)
<code>PCReg</code>	Sets a parameter indicating that the register is designated as a program counter. This attribute may only exist in a single register per instruction set. (Bool)
<code>Shared</code>	Sets a parameter indicating that the register is a shared register. The hardware generator will create ONE instance of the register for all cores. (Bool)
SubRegs	
<code>SubReg</code>	Sets the unique subregister name. The name must be unique across all registers within the register class (String)
<code>StartBit</code>	Sets the starting bit position of the subregister in the parent register. The start bit must be less than the end bit and a minimum of one bit. (Int)
<code>EndBit</code>	Sets the ending bit position of the subregister in the parent register. The end bit must be greater than the start bit and a minimum of one bit (Int)

```
1 Registers:
2   - RegName: String
3     Width: Integer
4     Index: Integer
5     PseudoName: String
6     IsFixedValue: Bool
7     IsSIMD: Bool
8     RWReg: Bool
9     ROReg: Bool
10    CSRReg: Bool
11    AMSReg: Bool
12    TUSReg: Bool
13    PCReg: Bool
14    Shared: Bool
15    SubRegs:
16      - SubReg: String
17        StartBit: Integer
18        EndBit: Integer
19      - SubReg: String
20        StartBit: Integer
21        EndBit: Integer
22  - RegName: String
23    Width: Integer
24    Index: Integer
25    PseudoName: String
26    IsFixedValue: Bool
27    IsSIMD: Bool
28    RWReg: Bool
29    ROReg: Bool
30    CSRReg: Bool
31    AMSReg: Bool
32    TUSReg: Bool
33    Shared: Bool
34    ...more registers
```

Listing 9: Register Node Definition

2.7 Register Classes

Register class nodes define an instance of a register file. The register file will contain some number of registers, each of which may have specific attributes. Indexing into the register file is defined by the registers contained therein and *does not* need to be monotonically increasing. Despite the ability to registers within the same register file that have disparate attributes, the HDL code generation mechanisms in CoreGen will select the largest bit width register in order to determine the space requirements of the register file. As a result, if the design requires several different sizes of registers, it is advantageous to separate them into distinct register files for the purpose of optimizing the eventual RTL.

Register classes are defined by two parameters: the register class name and the registers that exist within the register class. The registers defined in the register class are references to register defined in the register node section (See Section 2.6). The register naming convention must match that of the defined registers. We provide a sample register class IR block in Listing 10.

```
1 RegClasses:
2   - RegisterClassName: String
3     Registers:
4       - String
5       - String
6       - String
7   - RegisterClassName: String
8     Registers:
9       - String
10      - String
11      - String
12 ...more reg classes
```

Listing 10: Register Class Node Definition

2.8 Instruction Sets

Instruction set nodes are effectively container nodes. They are designated as a container for multiple instructions and instruction formats. As a result, the only parameter utilized to identify an individual instruction set is its respective name. Designs may have any number of instruction sets. We provide an example instruction set IR block in Listing 11.

```
1 ISAs :  
2   - ISAName: String  
3   - ISAName: String
```

Listing 11: Instruction Set Node Definition

2.9 Instruction Formats

The instruction format nodes define the formatting information, including field types, field widths and required information for each instruction format in **little endian**. The instruction format nodes do not define the actual instructions, only the encoding formats. Instruction formats can contain any number of instruction fields. The minimum field width is a single bit. The maximum field width is equivalent to the entire width of the instruction format (`FormatWidth`).

Each width must be designated as one of three potential field types. For register fields, you **must** also specify the target register class that is associated with the field (`RegClass`). Note that the width of the field should encapsulate the maximum potential index for the target register class. For instruction code fields, the width of the field will define the number of encodings.

Finally, instruction encodings are not required to contain contiguous fields. You may opt to leave undefined space between fields. The width of the encoding must match the largest ending bit field. For example, if your instruction format has the last field defined to end on bit 31 (`EndBit`), then your entire encoding must be at least 32 bits.

Table 6: CoreGen Instruction Format Node Parameters

Parameter	Description
ISA	Defines the instruction set architecture container. (String)
FormatWidth	Defines the width (in bits) of the register. (Int)
Fields	
FieldName	Defines the name of the respective field. (String)
FieldType	Defines the field type. This must be one of the following. (Special) CGInstReg : Defines a register field. Requires <code>RegClass</code> be defined. CGInstCode : Defines an instruction code (opcode, function code, etc). CGInstImm : Defines a field for an immediate value.
FieldWidth	Defines the width of the instruction field. (Integer)
StartBit	Defines the starting bit position of the field. (Integer)
EndBit	Defines the end bit position of the field. (Integer)
MandatoryField	Determines whether the field must be defined by the implementing instruction. (Bool)
RegClass	Sets the target register class parameter for <code>CGInstReg</code> fields. (String)

We provide an example Instruction Format IR block in Listing 12.

```

1 InstFormats:
2   - InstFormatName: String
3   ISA: String
4   FormatWidth: Integer
5   Fields:
6     - FieldName: String
7       FieldType: {CGInstReg, CGInstCode, CGInstImm}
8       FieldWidth: Integer
9       StartBit: Integer
10      EndBit: Integer
11      MandatoryField: Bool
12   - FieldName: String
13     FieldType: {CGInstReg, CGInstCode, CGInstImm}
14     FieldWidth: Integer

```

```
15     StartBit: Integer
16     EndBit: Integer
17     MandatoryField: Bool
18     RegClass: String
19     - FieldName: String
20     FieldType: {CGInstReg, CGInstCode, CGInstImm}
21     FieldWidth: Integer
22     StartBit: Integer
23     EndBit: Integer
24     MandatoryField: Bool
25     RegClass: String
26     - FieldName: String
27     FieldType: {CGInstReg, CGInstCode, CGInstImm}
28     FieldWidth: Integer
29     StartBit: Integer
30     EndBit: Integer
31     MandatoryField: Bool
32     RegClass: String
```

Listing 12: Instruction Format Node Definition

2.10 Instructions

Instruction nodes represent individual instances of single instructions. Each instruction node is attached to an instruction set reference as well as a specific instruction format. In addition to the aforementioned references, each instruction has the ability to include any necessary encoding information as well. A single instruction can define multiple encoding blocks, each with specific encoding information that is mapped back to an individual named field in the instruction format. For example, if the instruction format has a field called *opcode* that has the `MandatoryField` flag set to `true`, the instruction node may have an `Encoding` definition with the respective value for that instruction.

Note that it is up to the user to avoid collisions in the encodings across multiple instructions. The CoreGen tools may provide IR passes that check for these idiosyncracies, but the IR does not inherently prohibit collisions. Further, encoding definitions require that the user specify the encoding width (`EncodingWidth`). For opcode fields, this should be set to the same width as defined in the instruction encoding (`FieldWidth`). However, for fields such as immediate values, setting this width to something less than the full field width will automatically truncate the value to what is specified. For example, if the instruction encoding field for the immediate is 16 bits, but you define an instruction with a 12 bit immediate, the value in the immediate field of that instruction will be truncated to 12 bits.

Each instruction nodes can also drive the syntax by which the compiler's assembly code is generated and parsed. The `Syntax` field contains a string value in a specific format that defines the instruction mnemonics and associated syntax that is utilized to construct the associated assembler and disassembler. This syntax is constructed in a specific manner such that the CoreGen infrastructure has the ability to match the respective mnemonic to the correct instruction format field. The syntax must include the instruction name (which may differ from the instruction node name) and some number arguments. The argument names must match the respective register class or immediate field names of the respective instruction format. Register class arguments are delineated with `%` characters and immediate arguments are delineated with `$` characters. User may also intersperse commas and parenthesis throughout the syntax at will.

For example, consider an instruction `ADD` whose instruction format has three register class fields: `RT`, `RA` and `RB`. We may specify the instruction syntax as follows: `ADD %RT, %RA, %RB`. We may also have an additional instruction, `LOAD`, whose instruction format contains two register class fields (`RT`, `RA`) and a single immediate field (`IMM`). Our instruction syntax may then be formatted as: `LOAD %RT, $IMM(%RA)`. Note the use of the preceding characters for the register class and immediate field designators.

Finally, instructions have the ability to include inline `StoneCutter` definitions for each instruction (`NEED REFERENCE`). These inline `StoneCutter` instruction definitions are compiled consolidated into larger `StoneCutter` source files using the CoreGen infrastructure and subsequently compiled into Chisel HDL. Inline `StoneCutter` language is designated using `Impl` attributes within an individual instruction definition.

We provide an example instruction node in Listing 13.

```
1 Insts:  
2   - Inst: String  
3     ISA: String  
4     InstFormat: String  
5     Syntax: String  
6     Encodings:  
7       - EncodingField: String  
8         EncodingWidth: Integer  
9         EncodingValue: Integer  
10  - Inst: String  
11    ISA: String  
12    InstFormat: String  
13    Syntax: String  
14    Encodings:  
15      - EncodingField: String  
16        EncodingWidth: Integer  
17        EncodingValue: Integer  
18    Impl: String
```

Listing 13: Instruction Node Definition

2.11 Pseudo Instructions

Pseudo instruction nodes are special types of instruction nodes. They do not represent actual instructions. Rather, pseudo instructions contain special encoding directions for previously defined instructions and/or new instruction names. For example, in traditional RISC architectures register moves are implemented as unsigned addition instructions whereby the immediate values are set to zero. An example of this is noted as follows:

```
# register R0 = zero
add r10, r11, r0 # R10 = R11+R0
mov r10, r11     # Pseudo instruction for 'add r10, r11, r0'
```

Pseudo instructions require a unique name, the associated ISA and the target instruction that you seek to alias. You must also define a set of encoding parameters that are utilized to incorporate specific encodings for specific fields. For example, if you seek to define a pseudo instruction whereby an individual register file is always set to the same register index, utilize a pseudo instruction with an encoding field. Further, much like the original instruction definition, pseudo instructions permit the user to define assembly mnemonics that are echoed in the compiler implementation. The assembly mnemonics follow the syntax rules defined in Section 2.10. An example set of pseudo instructions is provided in Listing 14.

```
1 PseudoInsts:
2   - PseudoInst: String
3     ISA: String
4     Inst: String
5     Encodings:
6       - EncodingField: String
7         EncodingWidth: Integer
8         EncodingValue: Integer
9     Syntax: String
10  - PseudoInst: String
11    ISA: String
12    Inst: String
13    Encodings:
14      - EncodingField: String
15        EncodingWidth: Integer
16        EncodingValue: Integer
17    Syntax: String
```

Listing 14: Pseudo Instruction Node Definition

2.12 Caches

Cache nodes represent a single level of a traditional cache hierarchy. Regardless of whether the target use is designated as an instruction cache, data cache or otherwise, the cache node is defined in the same manner. The target use is determined by how the cache is connected to the core (or cores). Cache nodes are required to have at least two parameters: `Sets` and `Ways`. These represent the number of cache sets and the number of *ways* are represented in the cache.

Caches can also be configured in hierarchies. Each level in the cache must be defined as a cache node. To define a cache hierarchy, start with the cache layer closest to the core (usually defined as the *L1* layer) and add a `SubLevel` to the cache. The sublevel defines a connection to the next layer in the cache hierarchy. Nodes may exist as parents (sublevel to a lower level cache) and as a parent (contain a sublevel). You may define any number of caching layers providing that the connectivity is present between layers. An example caching hierarchy is providing in Listing 15.

```
1 Caches:
2   - Cache: String
3     Sets: Integer
4     Ways: Integer
5   - Cache: String
6     Sets: Integer
7     Ways: Integer
8     SubLevel: String
9   - Cache: String
10    Sets: Integer
11    Ways: Integer
```

Listing 15: Cache Node Definition

2.13 Scratchpad Memories

Scratchpad nodes are utilized to represent special case memory nodes that reside on die (as opposed to external memories such as DRAM). Further, these memory nodes participate in the normal addressing mechanisms, unlike cache hierarchies which are implied as a part of the memory pipeline.

Scratchpad nodes can be unique for individual cores or attached to multiple cores or other nodes. Each scratchpad memory node must include the size of the scratchpad (`MemSize`), the number of request ports (`RqstPorts`), the number of response ports (`RspPorts`) and the starting address (`StartAddr`). The total address space provided by the individual scratchpad is $StartAddr + MemSize$. Note that individual scratchpads may have the same address space providing that they are not connected to the same core. Further, multiple scratchpads can be connected to individual cores providing that the address spaces do not collide. An example scratchpad node definition is provided in Listing 16.

```
1 Scratchpads:
2   - Scratchpad: String
3     MemSize: Integer
4     RqstPorts: Integer
5     RspPorts: Integer
6     StartAddr: Integer
7   - Scratchpad: String
8     MemSize: Integer
9     RqstPorts: Integer
10    RspPorts: Integer
11    StartAddr: Integer
```

Listing 16: Scratchpad Node Definition

2.14 Virtual to Physical Translation Units

The virtual to physical (*VTP*) nodes include the ability to perform virtual to physical memory address translation in conjunction with the backend memory controller and scratchpad memory nodes. Currently, the VTP nodes require a single parameter, the name of the VTP node. Note that it may be inherently dangerous to have multiple VTP nodes in a single design without sufficient separation. For example, a heterogeneous design that contains multiple, tightly integrated cores may still have a single VTP node. However, a design that includes a loosely connected accelerator (with its own constituent memory controller) may perform its own virtual to physical translation. An example of the VTP node syntax is shown in Listing 17.

```
1 VTPControllers:  
2   - VTP: String
```

Listing 17: VTP Node Definition

2.15 Memory Controllers

Memory controller nodes are designed to receive physical addresses from upstream devices. The memory controller subsequently transfer the requests to the external memory devices. Many designs may warrant overriding these nodes with external RTL IP given design-specific external memory requirements. Otherwise, memory controller nodes require a unique node name as well as the number of upstream memory ports. Currently, the number of ports (`Ports`) should be an even number as these are divided evenly between request ports and response ports. An example memory controller IR block is shown in Listing 18.

```
1 MemoryControllers:  
2   - MemoryController: String  
3     Ports: Integer  
4   - MemoryController: String  
5     Ports: Integer
```

Listing 18: Memory Controller Node Definition

2.16 Communication Channels

Communication nodes provide connectivity between disparate nodes that would otherwise have no implied connectivity. For example, if you connect a cache node to a core, there is an implied link via the memory pipeline. However, if you connect a set of cores together (via a bus or other topology), there is no implication of how they may communicate. Communication nodes are utilized to connect various nodes or groups of nodes together into a singular topology. Nodes may include connectivity between multiple communication nodes in different networks. For example, a core may include connectivity to four adjacent cores in order to form a on-chip mesh interconnect.

Communication nodes require four main parameters as noted in Table 7. We show an example Communication node in Listing 19.

Table 7: CoreGen Instruction Format Node Parameters

Parameter	Description
Comm	Defines the unique name of the communication node. (String)
Type	Defines the communication node type. This must be one of the following. (Special) P2P : Defines a peer-to-peer communication channel for two node endpoints. Bus : Defines a bus communication channel with min 2 endpoints and no maximum. NOC : Defines a network on chip communication channel with min 2 endpoints and no maximum. Unknown : User-defined communication channel that requires user-provided RTL.
Endpoints	This contains a sequenced list of endpoint nodes that are connected to the communication channel.

```

1 Comms:
2   - Comm: String
3     Type: {P2P, Bus, NOC, Unknown}
4     Width: Integer
5     Endpoints:
6       - String
7       - String
8       - String

```

Listing 19: Communication Node Definition

2.17 Cores

Core nodes define a single instance of a traditional "core". This includes arithmetic facilities, register files and any required encoding data (instruction formats, register indices, etc). Each core node must include an associated instruction set definition (`ISA`), any required register classes for the target ISA (`RegisterClasses`) and optionally, one or more cache units. While caching units are optional dependent nodes for a core, we highly recommend that cores have at least an instruction cache facility. Otherwise, the instruction fetch performance of the core will be very poor.

In addition to the aforementioned base nodes, cores can be augmented with one or more extensions. These extensions may include additional, registers, register classes or other non-standard nodes beyond what is defined in the base core infrastructure.

Core nodes may also define the degree of symmetric multithreading (SMT) by specifying the number of `ThreadUnits`. This optional keyword permits users to specify the number of unique thread units for the respective core. The hardware code generator will output unique register files for each thread unit unless a register has been designated as *thread unit shared* (`TUSReg`) or `Shared`. See Section 2.6 for descriptions of these register attributes. The code generator will not, however, output unique `Extensions` for each thread unit. A single instance of a shared register will be generated and shared across the cores. If not specified, the number of `ThreadUnits` is assumed to be *1*.

We see an example of a basic and extended core node in Listing 20.

```

1 - Core: String
2   Cache: String
3   ISA: String
4   ThreadUnits: 2
5   RegisterClasses:
6     - RegClass: String
7     - RegClass: String
8 - Core: String
9   Cache: String
10  ISA: String
11  RegisterClasses:
12    - RegClass: String
13    - RegClass: String
14  Extensions:
15    - Extension: String

```

Listing 20: Core Node Definition

2.18 SoCs

SoC (or *System on Chip*) nodes are top-level, container nodes that are utilized to tie multiple cores and their dependent nodes together into a single package. Generally speaking, there should be only a single SoC node in a design. It is possible to have multiple SoC nodes in a given IR file, but the CoreGen pass infrastructure will likely flag this as being potentially erroneous. For each SoC node, you must define some number of cores that are attached. An example of this is shown in Listing 21.

```
1 Socs:  
2   - Soc: String  
3     Cores:  
4       - Core: String  
5       - Core: String
```

Listing 21: SoC Node Definition

2.19 Extensions

Extensions are unique container node types in CoreGen IR that represent nodes (or groups of nodes) that don't inherently fall into one of the other standard categories. Generally speaking, the node infrastructure within a given extension is not modified for a given design (although it can be modified if desired). An extension may fall into one of the following types:

- **Templates** : Template extensions are utilized to construct larger designs using basic building blocks of nodes. For example, a template extension may include a single core, L1 cache and communication link. This template extension could then be replicated multiple times within a design to build larger, multi-core SoC's.
- **Modules** : Module extensions are utilized to construct highly optimized hardware modules. Module extensions may include other node types, but often include user-provided RTL. Module extensions are analogous to "black box" IP that is re-used, but not modified.
- **Comms** : Communication extensions are utilized to construct special cases of templated extensions. These extensions are designed to construct more expressive communication infrastructures. For example, one may utilize a communication extension to construct a switching block for a custom network on chip architecture.
- **Unknown** : Unknown extensions are extensions that do not fall into the aforementioned types.

Given the containerized nature of extensions, one may include multiple types and hierarchies of other nodes within the extension. The extension may be very simple (single nodes) or arbitrarily complex. Extensions may include one or more of the following node types:

- Registers
- Register Classes
- ISAs
- Instruction Formats
- Instructions
- Pseudo Instructions
- Caches
- Cores
- Scratchpads
- Memory Controllers
- Communication Channels
- Extensions

We provide an example Extension IR block in Listing 22.

```
1 Extensions:
2   - Extension: String
3     Type: {Template, Module, Comm, Unknown}
4     Registers:
5       - RegName: String
6         Width: Integer
7         Index: Integer
8         IsFixedValue: Bool
9         IsSIMD: Bool
10        RWReg: Bool
11        ROReg: Bool
12        CSRReg: Bool
13        AMSReg: Bool
14        Shared: Bool
15    RegClasses:
16      - RegisterClassName: String
17        Registers:
18          - String
19    ISAs:
20      - ISAName: String
21    RTLFile: String
```

Listing 22: Extension Node Definition

2.20 Plugins

The final and most complex style of CoreGen IR node is the plugin node. These nodes contain both basic node data within the IR as well as the potential to trigger user-defined backend code generation mechanisms. Plugins are implemented as shared libraries that may contain one or more of the following:

- **Custom Node Features** : Plugin nodes have the ability to define an arbitrary set of features that can be modified by the user. Unlike *extensions* (Section 2.19) that may only include existing CoreGen node types as dependencies, plugins may include custom parameters. These parameters are defined in terms of their rudimentary types. We will describe this typing system below.
- **Custom Code Generation Mechanisms** : Given that each plugin is encapsulated in a shared library, the custom parameters defined above can be utilized to drive custom code generation mechanisms. This can be done in two ways. First, for existing RTL in the form of Chisel or Verilog, the custom parameter values can be utilized to drive custom CoreGen code generation output for the target plugin. Second, custom language backends can be utilized to output plugin-generated code in whatever form is necessary. However, be aware that any custom code generation facilities must adhere to the basic CoreGen naming conventions.
- **Custom Optimizations** : In addition to custom code generation mechanisms, plugins may also include custom optimizations. This includes plugin-specific interpretations of parameter values, dynamically generated child node dependencies and internal peephole optimizations.

For each plugin node defined in the IR, there are several required parameters that must be defined. First, each plugin must have a unique node name (`Plugin`). As is the case with other nodes, this parameter uniquely identifies the node with the dependence graph. Next, each plugin node must define its associated plugin library name (`PluginName`). This is the name of the plugin library utilized to implement the plugin. The `PluginName` *does not* contain any library prefix or file extensions. For example, if the target library was named *libSample.so*, the `PluginName` parameter would be *Sample*. Finally, the plugin node must initialize the major, minor and patch version numbers for the target plugin. In this manner, multiple versions of the same plugin can be installed on the development system and the CoreGen infrastructure will choose the correct version for the design. We see a full list of potential plugin parameters in Table 8.

Table 8: CoreGen Plugin Node Parameters

Parameter	Description
<code>Plugin</code>	Defines the target plugin unique <i>node</i> name. (String)
<code>PluginName</code>	Defines the name of the plugin library to load (no file extension). (String)
<code>MajorVersion</code>	Major version of the plugin to load. (Integer)
<code>MinorVersion</code>	Minor version of the plugin to load. (Integer)
<code>PatchVersion</code>	Patch version of the plugin to load. (Integer)
Features	
<code>FeatureName</code>	Name of the plugin-specific feature. (String)
<code>FeatureType</code>	Rudimentary type of the target feature (see list below). (Special)
<code>FeatureValue</code>	Value associated with the plugin-specific feature. (Special)

As mentioned above, each of the individual plugin nodes may include plugin-specific feature sets. Each feature must be defined and recognized by the plugin. The features are designated by the `FeatureName` and `FeatureType`. The permissible feature types are outlined in Table 9. For each `FeatureType`, you must also provide a `FeatureValue` parameter that defines the value of the respective feature in the respective data type. For example, if a feature is defined as a `Bool` type, then the value must be one of `true` or `false`.

In addition to the custom feature sets provided by plugin-specific logic, plugin nodes may also contain

Table 9: CoreGen Plugin Node Feature Types

FeatureType	Description
Unsigned	unsigned 32 bit integer (C++ <i>unsigned</i>)
UInt32t	unsigned 32 bit integer (C <i>uint32_t</i>)
Int32t	signed 32 bit integer (C <i>int32_t</i>)
UInt64t	unsigned 64 bit integer (C <i>uint64_t</i>)
Int64t	signed 64 bit integer (C <i>int64_t</i>)
Float	IEEE single precision floating point (C <i>float</i>)
Double	IEEE double precision floating point (C <i>double</i>)
String	String value (C++ <i>std::string</i>)
Bool	Boolean value (true or false)

dependent nodes in a similar manner as extensions. These nodes can be dynamically generated by the actual plugin or explicitly added to the plugin dependency graph. Plugins may define dependent nodes using the following node types:

- Registers
- Register Classes
- ISAs
- Instruction Formats
- Instructions
- Pseudo Instructions
- Caches
- Cores
- Scratchpads
- Memory Controllers
- Communication Channels
- Extensions
- SoC's

We provide a full example IR block for a plugin node in Listing 23

```

1 Plugins:
2   - Plugin: String
3     PluginName: String
4     MajorVersion: Integer
5     MinorVersion: Integer
6     PatchVersion: Integer
7     Features:
8       - FeatureName: String
9         FeatureType: Special
10        FeatureValue: Special

```

```
11     - FeatureName: String
12       FeatureType: Special
13       FeatureValue: Special
14 - Plugin: String
15   PluginName: String
16   MajorVersion: Integer
17   MinorVersion: Integer
18   PatchVersion: Integer
19   Features:
20     - FeatureName: String
21       FeatureType: Special
22       FeatureValue: Special
23     - FeatureName: String
24       FeatureType: Special
25       FeatureValue: Special
26   Registers:
27     - RegName: String
28       Width: Integer
29       Index: Integer
30       IsFixedValue: Bool
31       IsSIMD: Bool
32       RWReg: Bool
33       ROReg: Bool
34       CSRReg: Bool
35       AMSReg: Bool
36       Shared: Bool
37   RegClasses:
38     - RegisterClassName: String
39       Registers:
40         - String
41   ISAs:
42     - ISAName: String
```

Listing 23: Plugin Node Definition

3 Appendix A: Sample IR

```
1 ProjectInfo:
2   - ProjectName: TEST69
3     ProjectRoot: ./
4     ProjectType: unknown
5     ChiselMajorVersion: 3
6     ChiselMinorVersion: 0
7 Registers:
8   - RegName: TEST69.0.reg
9     Width: 64
10    Index: 0
11    PseudoName: pseudoreg.0
12    IsFixedValue: false
13    IsSIMD: false
14    RWReg: true
15    ROReg: false
16    CSRReg: true
17    AMSReg: false
18    Shared: false
19   - RegName: TEST69.1.reg
20     Width: 64
21     Index: 1
22     PseudoName: pseudoreg.1
23     IsFixedValue: false
24     IsSIMD: false
25     RWReg: true
26     ROReg: false
27     CSRReg: true
28     AMSReg: false
29     Shared: false
30   - RegName: TEST69.2.reg
31     Width: 64
32     Index: 2
33     PseudoName: pseudoreg.2
34     IsFixedValue: false
35     IsSIMD: false
36     RWReg: true
37     ROReg: false
38     CSRReg: true
39     AMSReg: false
40     Shared: false
41   - RegName: TEST69.3.reg
42     Width: 64
43     Index: 3
44     PseudoName: pseudoreg.3
45     IsFixedValue: false
46     IsSIMD: false
47     RWReg: true
48     ROReg: false
49     CSRReg: true
50     AMSReg: false
51     Shared: false
```

```
52 - RegName: TEST69.4.reg
53   Width: 64
54   Index: 4
55   PseudoName: pseudoreg.4
56   IsFixedValue: false
57   IsSIMD: false
58   RWReg: true
59   ROReg: false
60   CSRReg: true
61   AMSReg: false
62   Shared: false
63 - RegName: TEST69.5.reg
64   Width: 64
65   Index: 5
66   PseudoName: pseudoreg.5
67   IsFixedValue: false
68   IsSIMD: false
69   RWReg: true
70   ROReg: false
71   CSRReg: true
72   AMSReg: false
73   Shared: false
74 - RegName: TEST69.6.reg
75   Width: 64
76   Index: 6
77   PseudoName: pseudoreg.6
78   IsFixedValue: false
79   IsSIMD: false
80   RWReg: true
81   ROReg: false
82   CSRReg: true
83   AMSReg: false
84   Shared: false
85 - RegName: TEST69.7.reg
86   Width: 64
87   Index: 7
88   PseudoName: pseudoreg.7
89   IsFixedValue: false
90   IsSIMD: false
91   RWReg: true
92   ROReg: false
93   CSRReg: true
94   AMSReg: false
95   Shared: false
96 - RegName: TEST69.8.reg
97   Width: 64
98   Index: 8
99   PseudoName: pseudoreg.8
100  IsFixedValue: false
101  IsSIMD: false
102  RWReg: true
103  ROReg: false
104  CSRReg: true
105  AMSReg: false
```

```
106     Shared: false
107 - RegName: TEST69.9.reg
108     Width: 64
109     Index: 9
110     PseudoName: pseudoreg.9
111     IsFixedValue: false
112     IsSIMD: false
113     RWReg: true
114     ROReg: false
115     CSRReg: true
116     AMSReg: false
117     Shared: false
118 - RegName: TEST69.10.reg
119     Width: 64
120     Index: 10
121     PseudoName: pseudoreg.10
122     IsFixedValue: false
123     IsSIMD: false
124     RWReg: true
125     ROReg: false
126     CSRReg: true
127     AMSReg: false
128     Shared: false
129 - RegName: TEST69.11.reg
130     Width: 64
131     Index: 11
132     PseudoName: pseudoreg.11
133     IsFixedValue: false
134     IsSIMD: false
135     RWReg: true
136     ROReg: false
137     CSRReg: true
138     AMSReg: false
139     Shared: false
140 - RegName: TEST69.12.reg
141     Width: 64
142     Index: 12
143     PseudoName: pseudoreg.12
144     IsFixedValue: false
145     IsSIMD: false
146     RWReg: true
147     ROReg: false
148     CSRReg: true
149     AMSReg: false
150     Shared: false
151 - RegName: TEST69.13.reg
152     Width: 64
153     Index: 13
154     PseudoName: pseudoreg.13
155     IsFixedValue: false
156     IsSIMD: false
157     RWReg: true
158     ROReg: false
159     CSRReg: true
```



```
160     AMSReg: false
161     Shared: false
162     - RegName: TEST69.14.reg
163     Width: 64
164     Index: 14
165     PseudoName: pseudoreg.14
166     IsFixedValue: false
167     IsSIMD: false
168     RWReg: true
169     ROReg: false
170     CSRReg: true
171     AMSReg: false
172     Shared: false
173     - RegName: TEST69.15.reg
174     Width: 64
175     Index: 15
176     PseudoName: pseudoreg.15
177     IsFixedValue: false
178     IsSIMD: false
179     RWReg: true
180     ROReg: false
181     CSRReg: true
182     AMSReg: false
183     Shared: false
184     - RegName: TEST69.16.reg
185     Width: 64
186     Index: 16
187     PseudoName: pseudoreg.16
188     IsFixedValue: false
189     IsSIMD: false
190     RWReg: true
191     ROReg: false
192     CSRReg: true
193     AMSReg: false
194     Shared: false
195     - RegName: TEST69.17.reg
196     Width: 64
197     Index: 17
198     PseudoName: pseudoreg.17
199     IsFixedValue: false
200     IsSIMD: false
201     RWReg: true
202     ROReg: false
203     CSRReg: true
204     AMSReg: false
205     Shared: false
206     - RegName: TEST69.18.reg
207     Width: 64
208     Index: 18
209     PseudoName: pseudoreg.18
210     IsFixedValue: false
211     IsSIMD: false
212     RWReg: true
213     ROReg: false
```

```
214     CSRReg: true
215     AMSReg: false
216     Shared: false
217 -   RegName: TEST69.19.reg
218     Width: 64
219     Index: 19
220     PseudoName: pseudoreg.19
221     IsFixedValue: false
222     IsSIMD: false
223     RWReg: true
224     ROReg: false
225     CSRReg: true
226     AMSReg: false
227     Shared: false
228 -   RegName: TEST69.20.reg
229     Width: 64
230     Index: 20
231     PseudoName: pseudoreg.20
232     IsFixedValue: false
233     IsSIMD: false
234     RWReg: true
235     ROReg: false
236     CSRReg: true
237     AMSReg: false
238     Shared: false
239 -   RegName: TEST69.21.reg
240     Width: 64
241     Index: 21
242     PseudoName: pseudoreg.21
243     IsFixedValue: false
244     IsSIMD: false
245     RWReg: true
246     ROReg: false
247     CSRReg: true
248     AMSReg: false
249     Shared: false
250 -   RegName: TEST69.22.reg
251     Width: 64
252     Index: 22
253     PseudoName: pseudoreg.22
254     IsFixedValue: false
255     IsSIMD: false
256     RWReg: true
257     ROReg: false
258     CSRReg: true
259     AMSReg: false
260     Shared: false
261 -   RegName: TEST69.23.reg
262     Width: 64
263     Index: 23
264     PseudoName: pseudoreg.23
265     IsFixedValue: false
266     IsSIMD: false
267     RWReg: true
```

```
268     ROReg: false
269     CSRReg: true
270     AMSReg: false
271     Shared: false
272     - RegName: TEST69.24.reg
273     Width: 64
274     Index: 24
275     PseudoName: pseudoreg.24
276     IsFixedValue: false
277     IsSIMD: false
278     RWReg: true
279     ROReg: false
280     CSRReg: true
281     AMSReg: false
282     Shared: false
283     - RegName: TEST69.25.reg
284     Width: 64
285     Index: 25
286     PseudoName: pseudoreg.25
287     IsFixedValue: false
288     IsSIMD: false
289     RWReg: true
290     ROReg: false
291     CSRReg: true
292     AMSReg: false
293     Shared: false
294     - RegName: TEST69.26.reg
295     Width: 64
296     Index: 26
297     PseudoName: pseudoreg.26
298     IsFixedValue: false
299     IsSIMD: false
300     RWReg: true
301     ROReg: false
302     CSRReg: true
303     AMSReg: false
304     Shared: false
305     - RegName: TEST69.27.reg
306     Width: 64
307     Index: 27
308     PseudoName: pseudoreg.27
309     IsFixedValue: false
310     IsSIMD: false
311     RWReg: true
312     ROReg: false
313     CSRReg: true
314     AMSReg: false
315     Shared: false
316     - RegName: TEST69.28.reg
317     Width: 64
318     Index: 28
319     PseudoName: pseudoreg.28
320     IsFixedValue: false
321     IsSIMD: false
```

```
322 RWReg: true
323 ROReg: false
324 CSRReg: true
325 AMSReg: false
326 Shared: false
327 - RegName: TEST69.29.reg
328 Width: 64
329 Index: 29
330 PseudoName: pseudoreg.29
331 IsFixedValue: false
332 IsSIMD: false
333 RWReg: true
334 ROReg: false
335 CSRReg: true
336 AMSReg: false
337 Shared: false
338 - RegName: TEST69.30.reg
339 Width: 64
340 Index: 30
341 PseudoName: pseudoreg.30
342 IsFixedValue: false
343 IsSIMD: false
344 RWReg: true
345 ROReg: false
346 CSRReg: true
347 AMSReg: false
348 Shared: false
349 - RegName: TEST69.31.reg
350 Width: 64
351 Index: 31
352 PseudoName: pseudoreg.31
353 IsFixedValue: false
354 IsSIMD: false
355 RWReg: true
356 ROReg: false
357 CSRReg: true
358 AMSReg: false
359 Shared: false
360 - RegName: TEST69.32.reg
361 Width: 64
362 Index: 32
363 PseudoName: pseudoreg.32
364 IsFixedValue: false
365 IsSIMD: false
366 RWReg: true
367 ROReg: false
368 CSRReg: true
369 AMSReg: false
370 Shared: false
371 - RegName: TEST69.33.reg
372 Width: 64
373 Index: 33
374 PseudoName: pseudoreg.33
375 IsFixedValue: false
```

```
376     IsSIMD: false
377     RWReg: true
378     ROReg: false
379     CSRReg: true
380     AMSReg: false
381     Shared: false
382 -   RegName: TEST69.34.reg
383     Width: 64
384     Index: 34
385     PseudoName: pseudoreg.34
386     IsFixedValue: false
387     IsSIMD: false
388     RWReg: true
389     ROReg: false
390     CSRReg: true
391     AMSReg: false
392     Shared: false
393 -   RegName: TEST69.35.reg
394     Width: 64
395     Index: 35
396     PseudoName: pseudoreg.35
397     IsFixedValue: false
398     IsSIMD: false
399     RWReg: true
400     ROReg: false
401     CSRReg: true
402     AMSReg: false
403     Shared: false
404 -   RegName: TEST69.36.reg
405     Width: 64
406     Index: 36
407     PseudoName: pseudoreg.36
408     IsFixedValue: false
409     IsSIMD: false
410     RWReg: true
411     ROReg: false
412     CSRReg: true
413     AMSReg: false
414     Shared: false
415 -   RegName: TEST69.37.reg
416     Width: 64
417     Index: 37
418     PseudoName: pseudoreg.37
419     IsFixedValue: false
420     IsSIMD: false
421     RWReg: true
422     ROReg: false
423     CSRReg: true
424     AMSReg: false
425     Shared: false
426 -   RegName: TEST69.38.reg
427     Width: 64
428     Index: 38
429     PseudoName: pseudoreg.38
```

```
430     IsFixedValue: false
431     IsSIMD: false
432     RWReg: true
433     ROReg: false
434     CSRReg: true
435     AMSReg: false
436     Shared: false
437 - RegName: TEST69.39.reg
438     Width: 64
439     Index: 39
440     PseudoName: pseudoreg.39
441     IsFixedValue: false
442     IsSIMD: false
443     RWReg: true
444     ROReg: false
445     CSRReg: true
446     AMSReg: false
447     Shared: false
448 - RegName: TEST69.40.reg
449     Width: 64
450     Index: 40
451     PseudoName: pseudoreg.40
452     IsFixedValue: false
453     IsSIMD: false
454     RWReg: true
455     ROReg: false
456     CSRReg: true
457     AMSReg: false
458     Shared: false
459 - RegName: TEST69.41.reg
460     Width: 64
461     Index: 41
462     PseudoName: pseudoreg.41
463     IsFixedValue: false
464     IsSIMD: false
465     RWReg: true
466     ROReg: false
467     CSRReg: true
468     AMSReg: false
469     Shared: false
470 - RegName: TEST69.42.reg
471     Width: 64
472     Index: 42
473     PseudoName: pseudoreg.42
474     IsFixedValue: false
475     IsSIMD: false
476     RWReg: true
477     ROReg: false
478     CSRReg: true
479     AMSReg: false
480     Shared: false
481 - RegName: TEST69.43.reg
482     Width: 64
483     Index: 43
```

```
484 PseudoName: pseudoreg.43
485 IsFixedValue: false
486 IsSIMD: false
487 RWReg: true
488 ROReg: false
489 CSRReg: true
490 AMSReg: false
491 Shared: false
492 - RegName: TEST69.44.reg
493   Width: 64
494   Index: 44
495   PseudoName: pseudoreg.44
496   IsFixedValue: false
497   IsSIMD: false
498   RWReg: true
499   ROReg: false
500   CSRReg: true
501   AMSReg: false
502   Shared: false
503 - RegName: TEST69.45.reg
504   Width: 64
505   Index: 45
506   PseudoName: pseudoreg.45
507   IsFixedValue: false
508   IsSIMD: false
509   RWReg: true
510   ROReg: false
511   CSRReg: true
512   AMSReg: false
513   Shared: false
514 - RegName: TEST69.46.reg
515   Width: 64
516   Index: 46
517   PseudoName: pseudoreg.46
518   IsFixedValue: false
519   IsSIMD: false
520   RWReg: true
521   ROReg: false
522   CSRReg: true
523   AMSReg: false
524   Shared: false
525 - RegName: TEST69.47.reg
526   Width: 64
527   Index: 47
528   PseudoName: pseudoreg.47
529   IsFixedValue: false
530   IsSIMD: false
531   RWReg: true
532   ROReg: false
533   CSRReg: true
534   AMSReg: false
535   Shared: false
536 - RegName: TEST69.48.reg
537   Width: 64
```

```
538     Index: 48
539     PseudoName: pseudoreg.48
540     IsFixedValue: false
541     IsSIMD: false
542     RWReg: true
543     ROReg: false
544     CSRReg: true
545     AMSReg: false
546     Shared: false
547 -   RegName: TEST69.49.reg
548     Width: 64
549     Index: 49
550     PseudoName: pseudoreg.49
551     IsFixedValue: false
552     IsSIMD: false
553     RWReg: true
554     ROReg: false
555     CSRReg: true
556     AMSReg: false
557     Shared: false
558 -   RegName: TEST69.50.reg
559     Width: 64
560     Index: 50
561     PseudoName: pseudoreg.50
562     IsFixedValue: false
563     IsSIMD: false
564     RWReg: true
565     ROReg: false
566     CSRReg: true
567     AMSReg: false
568     Shared: false
569 -   RegName: TEST69.51.reg
570     Width: 64
571     Index: 51
572     PseudoName: pseudoreg.51
573     IsFixedValue: false
574     IsSIMD: false
575     RWReg: true
576     ROReg: false
577     CSRReg: true
578     AMSReg: false
579     Shared: false
580 -   RegName: TEST69.52.reg
581     Width: 64
582     Index: 52
583     PseudoName: pseudoreg.52
584     IsFixedValue: false
585     IsSIMD: false
586     RWReg: true
587     ROReg: false
588     CSRReg: true
589     AMSReg: false
590     Shared: false
591 -   RegName: TEST69.53.reg
```



```
592 Width: 64
593 Index: 53
594 PseudoName: pseudoreg.53
595 IsFixedValue: false
596 IsSIMD: false
597 RWReg: true
598 ROReg: false
599 CSRReg: true
600 AMSReg: false
601 Shared: false
602 - RegName: TEST69.54.reg
603 Width: 64
604 Index: 54
605 PseudoName: pseudoreg.54
606 IsFixedValue: false
607 IsSIMD: false
608 RWReg: true
609 ROReg: false
610 CSRReg: true
611 AMSReg: false
612 Shared: false
613 - RegName: TEST69.55.reg
614 Width: 64
615 Index: 55
616 PseudoName: pseudoreg.55
617 IsFixedValue: false
618 IsSIMD: false
619 RWReg: true
620 ROReg: false
621 CSRReg: true
622 AMSReg: false
623 Shared: false
624 - RegName: TEST69.56.reg
625 Width: 64
626 Index: 56
627 PseudoName: pseudoreg.56
628 IsFixedValue: false
629 IsSIMD: false
630 RWReg: true
631 ROReg: false
632 CSRReg: true
633 AMSReg: false
634 Shared: false
635 - RegName: TEST69.57.reg
636 Width: 64
637 Index: 57
638 PseudoName: pseudoreg.57
639 IsFixedValue: false
640 IsSIMD: false
641 RWReg: true
642 ROReg: false
643 CSRReg: true
644 AMSReg: false
645 Shared: false
```

```
646 - RegName: TEST69.58.reg
647   Width: 64
648   Index: 58
649   PseudoName: pseudoreg.58
650   IsFixedValue: false
651   IsSIMD: false
652   RWReg: true
653   ROReg: false
654   CSRReg: true
655   AMSReg: false
656   Shared: false
657 - RegName: TEST69.59.reg
658   Width: 64
659   Index: 59
660   PseudoName: pseudoreg.59
661   IsFixedValue: false
662   IsSIMD: false
663   RWReg: true
664   ROReg: false
665   CSRReg: true
666   AMSReg: false
667   Shared: false
668 - RegName: TEST69.60.reg
669   Width: 64
670   Index: 60
671   PseudoName: pseudoreg.60
672   IsFixedValue: false
673   IsSIMD: false
674   RWReg: true
675   ROReg: false
676   CSRReg: true
677   AMSReg: false
678   Shared: false
679 - RegName: TEST69.61.reg
680   Width: 64
681   Index: 61
682   PseudoName: pseudoreg.61
683   IsFixedValue: false
684   IsSIMD: false
685   RWReg: true
686   ROReg: false
687   CSRReg: true
688   AMSReg: false
689   Shared: false
690 - RegName: TEST69.62.reg
691   Width: 64
692   Index: 62
693   PseudoName: pseudoreg.62
694   IsFixedValue: false
695   IsSIMD: false
696   RWReg: true
697   ROReg: false
698   CSRReg: true
699   AMSReg: false
```

```
700     Shared: false
701 - RegName: TEST69.63.reg
702     Width: 64
703     Index: 63
704     PseudoName: pseudoreg.63
705     IsFixedValue: false
706     IsSIMD: false
707     RWReg: true
708     ROReg: false
709     CSRReg: true
710     AMSReg: false
711     Shared: false
712 - RegName: TEST69.64.reg
713     Width: 64
714     Index: 64
715     PseudoName: pseudoreg.64
716     IsFixedValue: false
717     IsSIMD: false
718     RWReg: true
719     ROReg: false
720     CSRReg: true
721     AMSReg: false
722     Shared: false
723 - RegName: TEST69.65.reg
724     Width: 64
725     Index: 65
726     PseudoName: pseudoreg.65
727     IsFixedValue: false
728     IsSIMD: false
729     RWReg: true
730     ROReg: false
731     CSRReg: true
732     AMSReg: false
733     Shared: false
734 - RegName: TEST69.66.reg
735     Width: 64
736     Index: 66
737     PseudoName: pseudoreg.66
738     IsFixedValue: false
739     IsSIMD: false
740     RWReg: true
741     ROReg: false
742     CSRReg: true
743     AMSReg: false
744     Shared: false
745 - RegName: TEST69.67.reg
746     Width: 64
747     Index: 67
748     PseudoName: pseudoreg.67
749     IsFixedValue: false
750     IsSIMD: false
751     RWReg: true
752     ROReg: false
753     CSRReg: true
```

```
754     AMSReg: false
755     Shared: false
756 -   RegName: TEST69.68.reg
757     Width: 64
758     Index: 68
759     PseudoName: pseudoreg.68
760     IsFixedValue: false
761     IsSIMD: false
762     RWReg: true
763     ROReg: false
764     CSRReg: true
765     AMSReg: false
766     Shared: false
767 -   RegName: TEST69.0.csr
768     Width: 64
769     Index: 0
770     PseudoName: csr.0
771     IsFixedValue: false
772     IsSIMD: false
773     RWReg: false
774     ROReg: false
775     CSRReg: true
776     AMSReg: false
777     Shared: false
778 -   RegName: TEST69.1.csr
779     Width: 64
780     Index: 1
781     PseudoName: csr.1
782     IsFixedValue: false
783     IsSIMD: false
784     RWReg: false
785     ROReg: false
786     CSRReg: true
787     AMSReg: false
788     Shared: false
789 -   RegName: TEST69.2.csr
790     Width: 64
791     Index: 2
792     PseudoName: csr.2
793     IsFixedValue: false
794     IsSIMD: false
795     RWReg: false
796     ROReg: false
797     CSRReg: true
798     AMSReg: false
799     Shared: false
800 -   RegName: TEST69.3.csr
801     Width: 64
802     Index: 3
803     PseudoName: csr.3
804     IsFixedValue: false
805     IsSIMD: false
806     RWReg: false
807     ROReg: false
```

```
808     CSRReg: true
809     AMSReg: false
810     Shared: false
811 -   RegName: TEST69.4.csr
812     Width: 64
813     Index: 4
814     PseudoName: csr.4
815     IsFixedValue: false
816     IsSIMD: false
817     RWReg: false
818     ROReg: false
819     CSRReg: true
820     AMSReg: false
821     Shared: false
822 -   RegName: TEST69.5.csr
823     Width: 64
824     Index: 5
825     PseudoName: csr.5
826     IsFixedValue: false
827     IsSIMD: false
828     RWReg: false
829     ROReg: false
830     CSRReg: true
831     AMSReg: false
832     Shared: false
833 -   RegName: TEST69.6.csr
834     Width: 64
835     Index: 6
836     PseudoName: csr.6
837     IsFixedValue: false
838     IsSIMD: false
839     RWReg: false
840     ROReg: false
841     CSRReg: true
842     AMSReg: false
843     Shared: false
844 -   RegName: TEST69.7.csr
845     Width: 64
846     Index: 7
847     PseudoName: csr.7
848     IsFixedValue: false
849     IsSIMD: false
850     RWReg: false
851     ROReg: false
852     CSRReg: true
853     AMSReg: false
854     Shared: false
855 -   RegName: TEST69.8.csr
856     Width: 64
857     Index: 8
858     PseudoName: csr.8
859     IsFixedValue: false
860     IsSIMD: false
861     RWReg: false
```

```
862     ROReg: false
863     CSRReg: true
864     AMSReg: false
865     Shared: false
866     - RegName: TEST69.9.csr
867     Width: 64
868     Index: 9
869     PseudoName: csr.9
870     IsFixedValue: false
871     IsSIMD: false
872     RWReg: false
873     ROReg: false
874     CSRReg: true
875     AMSReg: false
876     Shared: false
877     - RegName: TEST69.10.csr
878     Width: 64
879     Index: 10
880     PseudoName: csr.10
881     IsFixedValue: false
882     IsSIMD: false
883     RWReg: false
884     ROReg: false
885     CSRReg: true
886     AMSReg: false
887     Shared: false
888     - RegName: TEST69.11.csr
889     Width: 64
890     Index: 11
891     PseudoName: csr.11
892     IsFixedValue: false
893     IsSIMD: false
894     RWReg: false
895     ROReg: false
896     CSRReg: true
897     AMSReg: false
898     Shared: false
899     - RegName: TEST69.12.csr
900     Width: 64
901     Index: 12
902     PseudoName: csr.12
903     IsFixedValue: false
904     IsSIMD: false
905     RWReg: false
906     ROReg: false
907     CSRReg: true
908     AMSReg: false
909     Shared: false
910     - RegName: TEST69.13.csr
911     Width: 64
912     Index: 13
913     PseudoName: csr.13
914     IsFixedValue: false
915     IsSIMD: false
```

```
916 RWReg: false
917 ROReg: false
918 CSRReg: true
919 AMSReg: false
920 Shared: false
921 - RegName: TEST69.14.csr
922 Width: 64
923 Index: 14
924 PseudoName: csr.14
925 IsFixedValue: false
926 IsSIMD: false
927 RWReg: false
928 ROReg: false
929 CSRReg: true
930 AMSReg: false
931 Shared: false
932 - RegName: TEST69.15.csr
933 Width: 64
934 Index: 15
935 PseudoName: csr.15
936 IsFixedValue: false
937 IsSIMD: false
938 RWReg: false
939 ROReg: false
940 CSRReg: true
941 AMSReg: false
942 Shared: false
943 - RegName: TEST69.16.csr
944 Width: 64
945 Index: 16
946 PseudoName: csr.16
947 IsFixedValue: false
948 IsSIMD: false
949 RWReg: false
950 ROReg: false
951 CSRReg: true
952 AMSReg: false
953 Shared: false
954 - RegName: TEST69.17.csr
955 Width: 64
956 Index: 17
957 PseudoName: csr.17
958 IsFixedValue: false
959 IsSIMD: false
960 RWReg: false
961 ROReg: false
962 CSRReg: true
963 AMSReg: false
964 Shared: false
965 - RegName: TEST69.18.csr
966 Width: 64
967 Index: 18
968 PseudoName: csr.18
969 IsFixedValue: false
```

```
970     IsSIMD: false
971     RWReg: false
972     ROReg: false
973     CSRReg: true
974     AMSReg: false
975     Shared: false
976 -   RegName: TEST69.19.csr
977     Width: 64
978     Index: 19
979     PseudoName: csr.19
980     IsFixedValue: false
981     IsSIMD: false
982     RWReg: false
983     ROReg: false
984     CSRReg: true
985     AMSReg: false
986     Shared: false
987 -   RegName: TEST69.20.csr
988     Width: 64
989     Index: 20
990     PseudoName: csr.20
991     IsFixedValue: false
992     IsSIMD: false
993     RWReg: false
994     ROReg: false
995     CSRReg: true
996     AMSReg: false
997     Shared: false
998 -   RegName: TEST69.21.csr
999     Width: 64
1000    Index: 21
1001    PseudoName: csr.21
1002    IsFixedValue: false
1003    IsSIMD: false
1004    RWReg: false
1005    ROReg: false
1006    CSRReg: true
1007    AMSReg: false
1008    Shared: false
1009 -   RegName: TEST69.22.csr
1010    Width: 64
1011    Index: 22
1012    PseudoName: csr.22
1013    IsFixedValue: false
1014    IsSIMD: false
1015    RWReg: false
1016    ROReg: false
1017    CSRReg: true
1018    AMSReg: false
1019    Shared: false
1020 -   RegName: TEST69.23.csr
1021    Width: 64
1022    Index: 23
1023    PseudoName: csr.23
```



```
1024     IsFixedValue: false
1025     IsSIMD: false
1026     RWReg: false
1027     ROReg: false
1028     CSRReg: true
1029     AMSReg: false
1030     Shared: false
1031 - RegName: TEST69.24.csr
1032     Width: 64
1033     Index: 24
1034     PseudoName: csr.24
1035     IsFixedValue: false
1036     IsSIMD: false
1037     RWReg: false
1038     ROReg: false
1039     CSRReg: true
1040     AMSReg: false
1041     Shared: false
1042 - RegName: TEST69.25.csr
1043     Width: 64
1044     Index: 25
1045     PseudoName: csr.25
1046     IsFixedValue: false
1047     IsSIMD: false
1048     RWReg: false
1049     ROReg: false
1050     CSRReg: true
1051     AMSReg: false
1052     Shared: false
1053 - RegName: TEST69.26.csr
1054     Width: 64
1055     Index: 26
1056     PseudoName: csr.26
1057     IsFixedValue: false
1058     IsSIMD: false
1059     RWReg: false
1060     ROReg: false
1061     CSRReg: true
1062     AMSReg: false
1063     Shared: false
1064 - RegName: TEST69.27.csr
1065     Width: 64
1066     Index: 27
1067     PseudoName: csr.27
1068     IsFixedValue: false
1069     IsSIMD: false
1070     RWReg: false
1071     ROReg: false
1072     CSRReg: true
1073     AMSReg: false
1074     Shared: false
1075 - RegName: TEST69.28.csr
1076     Width: 64
1077     Index: 28
```

```
1078 PseudoName: csr.28
1079 IsFixedValue: false
1080 IsSIMD: false
1081 RWReg: false
1082 ROReg: false
1083 CSRReg: true
1084 AMSReg: false
1085 Shared: false
1086 - RegName: TEST69.29.csr
1087 Width: 64
1088 Index: 29
1089 PseudoName: csr.29
1090 IsFixedValue: false
1091 IsSIMD: false
1092 RWReg: false
1093 ROReg: false
1094 CSRReg: true
1095 AMSReg: false
1096 Shared: false
1097 - RegName: TEST69.30.csr
1098 Width: 64
1099 Index: 30
1100 PseudoName: csr.30
1101 IsFixedValue: false
1102 IsSIMD: false
1103 RWReg: false
1104 ROReg: false
1105 CSRReg: true
1106 AMSReg: false
1107 Shared: false
1108 - RegName: TEST69.31.csr
1109 Width: 64
1110 Index: 31
1111 PseudoName: csr.31
1112 IsFixedValue: false
1113 IsSIMD: false
1114 RWReg: false
1115 ROReg: false
1116 CSRReg: true
1117 AMSReg: false
1118 Shared: false
1119 - RegName: TEST69.32.csr
1120 Width: 64
1121 Index: 32
1122 PseudoName: csr.32
1123 IsFixedValue: false
1124 IsSIMD: false
1125 RWReg: false
1126 ROReg: false
1127 CSRReg: true
1128 AMSReg: false
1129 Shared: false
1130 - RegName: TEST69.33.csr
1131 Width: 64
```

```
1132     Index: 33
1133     PseudoName: csr.33
1134     IsFixedValue: false
1135     IsSIMD: false
1136     RWReg: false
1137     ROReg: false
1138     CSRReg: true
1139     AMSReg: false
1140     Shared: false
1141     - RegName: TEST69.34.csr
1142     Width: 64
1143     Index: 34
1144     PseudoName: csr.34
1145     IsFixedValue: false
1146     IsSIMD: false
1147     RWReg: false
1148     ROReg: false
1149     CSRReg: true
1150     AMSReg: false
1151     Shared: false
1152     - RegName: TEST69.35.csr
1153     Width: 64
1154     Index: 35
1155     PseudoName: csr.35
1156     IsFixedValue: false
1157     IsSIMD: false
1158     RWReg: false
1159     ROReg: false
1160     CSRReg: true
1161     AMSReg: false
1162     Shared: false
1163     - RegName: TEST69.36.csr
1164     Width: 64
1165     Index: 36
1166     PseudoName: csr.36
1167     IsFixedValue: false
1168     IsSIMD: false
1169     RWReg: false
1170     ROReg: false
1171     CSRReg: true
1172     AMSReg: false
1173     Shared: false
1174     - RegName: TEST69.37.csr
1175     Width: 64
1176     Index: 37
1177     PseudoName: csr.37
1178     IsFixedValue: false
1179     IsSIMD: false
1180     RWReg: false
1181     ROReg: false
1182     CSRReg: true
1183     AMSReg: false
1184     Shared: false
1185     - RegName: TEST69.38.csr
```

```
1186 Width: 64
1187 Index: 38
1188 PseudoName: csr.38
1189 IsFixedValue: false
1190 IsSIMD: false
1191 RWReg: false
1192 ROReg: false
1193 CSRReg: true
1194 AMSReg: false
1195 Shared: false
1196 - RegName: TEST69.39.csr
1197 Width: 64
1198 Index: 39
1199 PseudoName: csr.39
1200 IsFixedValue: false
1201 IsSIMD: false
1202 RWReg: false
1203 ROReg: false
1204 CSRReg: true
1205 AMSReg: false
1206 Shared: false
1207 - RegName: TEST69.40.csr
1208 Width: 64
1209 Index: 40
1210 PseudoName: csr.40
1211 IsFixedValue: false
1212 IsSIMD: false
1213 RWReg: false
1214 ROReg: false
1215 CSRReg: true
1216 AMSReg: false
1217 Shared: false
1218 - RegName: TEST69.41.csr
1219 Width: 64
1220 Index: 41
1221 PseudoName: csr.41
1222 IsFixedValue: false
1223 IsSIMD: false
1224 RWReg: false
1225 ROReg: false
1226 CSRReg: true
1227 AMSReg: false
1228 Shared: false
1229 - RegName: TEST69.42.csr
1230 Width: 64
1231 Index: 42
1232 PseudoName: csr.42
1233 IsFixedValue: false
1234 IsSIMD: false
1235 RWReg: false
1236 ROReg: false
1237 CSRReg: true
1238 AMSReg: false
1239 Shared: false
```

```
1240 - RegName: TEST69.43.csr
1241   Width: 64
1242   Index: 43
1243   PseudoName: csr.43
1244   IsFixedValue: false
1245   IsSIMD: false
1246   RWReg: false
1247   ROReg: false
1248   CSRReg: true
1249   AMSReg: false
1250   Shared: false
1251 - RegName: TEST69.44.csr
1252   Width: 64
1253   Index: 44
1254   PseudoName: csr.44
1255   IsFixedValue: false
1256   IsSIMD: false
1257   RWReg: false
1258   ROReg: false
1259   CSRReg: true
1260   AMSReg: false
1261   Shared: false
1262 - RegName: TEST69.45.csr
1263   Width: 64
1264   Index: 45
1265   PseudoName: csr.45
1266   IsFixedValue: false
1267   IsSIMD: false
1268   RWReg: false
1269   ROReg: false
1270   CSRReg: true
1271   AMSReg: false
1272   Shared: false
1273 - RegName: TEST69.46.csr
1274   Width: 64
1275   Index: 46
1276   PseudoName: csr.46
1277   IsFixedValue: false
1278   IsSIMD: false
1279   RWReg: false
1280   ROReg: false
1281   CSRReg: true
1282   AMSReg: false
1283   Shared: false
1284 - RegName: TEST69.47.csr
1285   Width: 64
1286   Index: 47
1287   PseudoName: csr.47
1288   IsFixedValue: false
1289   IsSIMD: false
1290   RWReg: false
1291   ROReg: false
1292   CSRReg: true
1293   AMSReg: false
```

```
1294     Shared: false
1295 - RegName: TEST69.48.csr
1296     Width: 64
1297     Index: 48
1298     PseudoName: csr.48
1299     IsFixedValue: false
1300     IsSIMD: false
1301     RWReg: false
1302     ROReg: false
1303     CSRReg: true
1304     AMSReg: false
1305     Shared: false
1306 - RegName: TEST69.49.csr
1307     Width: 64
1308     Index: 49
1309     PseudoName: csr.49
1310     IsFixedValue: false
1311     IsSIMD: false
1312     RWReg: false
1313     ROReg: false
1314     CSRReg: true
1315     AMSReg: false
1316     Shared: false
1317 - RegName: TEST69.50.csr
1318     Width: 64
1319     Index: 50
1320     PseudoName: csr.50
1321     IsFixedValue: false
1322     IsSIMD: false
1323     RWReg: false
1324     ROReg: false
1325     CSRReg: true
1326     AMSReg: false
1327     Shared: false
1328 - RegName: TEST69.51.csr
1329     Width: 64
1330     Index: 51
1331     PseudoName: csr.51
1332     IsFixedValue: false
1333     IsSIMD: false
1334     RWReg: false
1335     ROReg: false
1336     CSRReg: true
1337     AMSReg: false
1338     Shared: false
1339 - RegName: TEST69.52.csr
1340     Width: 64
1341     Index: 52
1342     PseudoName: csr.52
1343     IsFixedValue: false
1344     IsSIMD: false
1345     RWReg: false
1346     ROReg: false
1347     CSRReg: true
```

```
1348     AMSReg: false
1349     Shared: false
1350 -   RegName: TEST69.53.csr
1351     Width: 64
1352     Index: 53
1353     PseudoName: csr.53
1354     IsFixedValue: false
1355     IsSIMD: false
1356     RWReg: false
1357     ROReg: false
1358     CSRReg: true
1359     AMSReg: false
1360     Shared: false
1361 -   RegName: TEST69.54.csr
1362     Width: 64
1363     Index: 54
1364     PseudoName: csr.54
1365     IsFixedValue: false
1366     IsSIMD: false
1367     RWReg: false
1368     ROReg: false
1369     CSRReg: true
1370     AMSReg: false
1371     Shared: false
1372 -   RegName: TEST69.55.csr
1373     Width: 64
1374     Index: 55
1375     PseudoName: csr.55
1376     IsFixedValue: false
1377     IsSIMD: false
1378     RWReg: false
1379     ROReg: false
1380     CSRReg: true
1381     AMSReg: false
1382     Shared: false
1383 -   RegName: TEST69.56.csr
1384     Width: 64
1385     Index: 56
1386     PseudoName: csr.56
1387     IsFixedValue: false
1388     IsSIMD: false
1389     RWReg: false
1390     ROReg: false
1391     CSRReg: true
1392     AMSReg: false
1393     Shared: false
1394 -   RegName: TEST69.57.csr
1395     Width: 64
1396     Index: 57
1397     PseudoName: csr.57
1398     IsFixedValue: false
1399     IsSIMD: false
1400     RWReg: false
1401     ROReg: false
```

```
1402 CSRReg: true
1403 AMSReg: false
1404 Shared: false
1405 - RegName: TEST69.58.csr
1406 Width: 64
1407 Index: 58
1408 PseudoName: csr.58
1409 IsFixedValue: false
1410 IsSIMD: false
1411 RWReg: false
1412 ROReg: false
1413 CSRReg: true
1414 AMSReg: false
1415 Shared: false
1416 - RegName: TEST69.59.csr
1417 Width: 64
1418 Index: 59
1419 PseudoName: csr.59
1420 IsFixedValue: false
1421 IsSIMD: false
1422 RWReg: false
1423 ROReg: false
1424 CSRReg: true
1425 AMSReg: false
1426 Shared: false
1427 - RegName: TEST69.60.csr
1428 Width: 64
1429 Index: 60
1430 PseudoName: csr.60
1431 IsFixedValue: false
1432 IsSIMD: false
1433 RWReg: false
1434 ROReg: false
1435 CSRReg: true
1436 AMSReg: false
1437 Shared: false
1438 - RegName: TEST69.61.csr
1439 Width: 64
1440 Index: 61
1441 PseudoName: csr.61
1442 IsFixedValue: false
1443 IsSIMD: false
1444 RWReg: false
1445 ROReg: false
1446 CSRReg: true
1447 AMSReg: false
1448 Shared: false
1449 - RegName: TEST69.62.csr
1450 Width: 64
1451 Index: 62
1452 PseudoName: csr.62
1453 IsFixedValue: false
1454 IsSIMD: false
1455 RWReg: false
```



```
1456     ROReg: false
1457     CSRReg: true
1458     AMSReg: false
1459     Shared: false
1460 -   RegName: TEST69.63.csr
1461     Width: 64
1462     Index: 63
1463     PseudoName: csr.63
1464     IsFixedValue: false
1465     IsSIMD: false
1466     RWReg: false
1467     ROReg: false
1468     CSRReg: true
1469     AMSReg: false
1470     Shared: false
1471 -   RegName: TEST69.64.csr
1472     Width: 64
1473     Index: 64
1474     PseudoName: csr.64
1475     IsFixedValue: false
1476     IsSIMD: false
1477     RWReg: false
1478     ROReg: false
1479     CSRReg: true
1480     AMSReg: false
1481     Shared: false
1482 -   RegName: TEST69.65.csr
1483     Width: 64
1484     Index: 65
1485     PseudoName: csr.65
1486     IsFixedValue: false
1487     IsSIMD: false
1488     RWReg: false
1489     ROReg: false
1490     CSRReg: true
1491     AMSReg: false
1492     Shared: false
1493 -   RegName: TEST69.66.csr
1494     Width: 64
1495     Index: 66
1496     PseudoName: csr.66
1497     IsFixedValue: false
1498     IsSIMD: false
1499     RWReg: false
1500     ROReg: false
1501     CSRReg: true
1502     AMSReg: false
1503     Shared: false
1504 -   RegName: TEST69.67.csr
1505     Width: 64
1506     Index: 67
1507     PseudoName: csr.67
1508     IsFixedValue: false
1509     IsSIMD: false
```

```
1510 RWReg: false
1511 ROReg: false
1512 CSRReg: true
1513 AMSReg: false
1514 Shared: false
1515 - RegName: TEST69.68.csr
1516 Width: 64
1517 Index: 68
1518 PseudoName: csr.68
1519 IsFixedValue: false
1520 IsSIMD: false
1521 RWReg: false
1522 ROReg: false
1523 CSRReg: true
1524 AMSReg: false
1525 Shared: false
1526 RegClasses:
1527 - RegisterClassName: TEST69.regclass
1528 Registers:
1529 - TEST69.0.reg
1530 - TEST69.1.reg
1531 - TEST69.2.reg
1532 - TEST69.3.reg
1533 - TEST69.4.reg
1534 - TEST69.5.reg
1535 - TEST69.6.reg
1536 - TEST69.7.reg
1537 - TEST69.8.reg
1538 - TEST69.9.reg
1539 - TEST69.10.reg
1540 - TEST69.11.reg
1541 - TEST69.12.reg
1542 - TEST69.13.reg
1543 - TEST69.14.reg
1544 - TEST69.15.reg
1545 - TEST69.16.reg
1546 - TEST69.17.reg
1547 - TEST69.18.reg
1548 - TEST69.19.reg
1549 - TEST69.20.reg
1550 - TEST69.21.reg
1551 - TEST69.22.reg
1552 - TEST69.23.reg
1553 - TEST69.24.reg
1554 - TEST69.25.reg
1555 - TEST69.26.reg
1556 - TEST69.27.reg
1557 - TEST69.28.reg
1558 - TEST69.29.reg
1559 - TEST69.30.reg
1560 - TEST69.31.reg
1561 - TEST69.32.reg
1562 - TEST69.33.reg
1563 - TEST69.34.reg
```

```
1564 - TEST69.35.reg
1565 - TEST69.36.reg
1566 - TEST69.37.reg
1567 - TEST69.38.reg
1568 - TEST69.39.reg
1569 - TEST69.40.reg
1570 - TEST69.41.reg
1571 - TEST69.42.reg
1572 - TEST69.43.reg
1573 - TEST69.44.reg
1574 - TEST69.45.reg
1575 - TEST69.46.reg
1576 - TEST69.47.reg
1577 - TEST69.48.reg
1578 - TEST69.49.reg
1579 - TEST69.50.reg
1580 - TEST69.51.reg
1581 - TEST69.52.reg
1582 - TEST69.53.reg
1583 - TEST69.54.reg
1584 - TEST69.55.reg
1585 - TEST69.56.reg
1586 - TEST69.57.reg
1587 - TEST69.58.reg
1588 - TEST69.59.reg
1589 - TEST69.60.reg
1590 - TEST69.61.reg
1591 - TEST69.62.reg
1592 - TEST69.63.reg
1593 - TEST69.64.reg
1594 - TEST69.65.reg
1595 - TEST69.66.reg
1596 - TEST69.67.reg
1597 - TEST69.68.reg
1598 - RegisterClassName: TEST69.csrregclass
1599 Registers:
1600 - TEST69.0.csr
1601 - TEST69.1.csr
1602 - TEST69.2.csr
1603 - TEST69.3.csr
1604 - TEST69.4.csr
1605 - TEST69.5.csr
1606 - TEST69.6.csr
1607 - TEST69.7.csr
1608 - TEST69.8.csr
1609 - TEST69.9.csr
1610 - TEST69.10.csr
1611 - TEST69.11.csr
1612 - TEST69.12.csr
1613 - TEST69.13.csr
1614 - TEST69.14.csr
1615 - TEST69.15.csr
1616 - TEST69.16.csr
1617 - TEST69.17.csr
```

```
1618 - TEST69.18.csr
1619 - TEST69.19.csr
1620 - TEST69.20.csr
1621 - TEST69.21.csr
1622 - TEST69.22.csr
1623 - TEST69.23.csr
1624 - TEST69.24.csr
1625 - TEST69.25.csr
1626 - TEST69.26.csr
1627 - TEST69.27.csr
1628 - TEST69.28.csr
1629 - TEST69.29.csr
1630 - TEST69.30.csr
1631 - TEST69.31.csr
1632 - TEST69.32.csr
1633 - TEST69.33.csr
1634 - TEST69.34.csr
1635 - TEST69.35.csr
1636 - TEST69.36.csr
1637 - TEST69.37.csr
1638 - TEST69.38.csr
1639 - TEST69.39.csr
1640 - TEST69.40.csr
1641 - TEST69.41.csr
1642 - TEST69.42.csr
1643 - TEST69.43.csr
1644 - TEST69.44.csr
1645 - TEST69.45.csr
1646 - TEST69.46.csr
1647 - TEST69.47.csr
1648 - TEST69.48.csr
1649 - TEST69.49.csr
1650 - TEST69.50.csr
1651 - TEST69.51.csr
1652 - TEST69.52.csr
1653 - TEST69.53.csr
1654 - TEST69.54.csr
1655 - TEST69.55.csr
1656 - TEST69.56.csr
1657 - TEST69.57.csr
1658 - TEST69.58.csr
1659 - TEST69.59.csr
1660 - TEST69.60.csr
1661 - TEST69.61.csr
1662 - TEST69.62.csr
1663 - TEST69.63.csr
1664 - TEST69.64.csr
1665 - TEST69.65.csr
1666 - TEST69.66.csr
1667 - TEST69.67.csr
1668 - TEST69.68.csr
1669 ISAs:
1670 - ISAName: TEST69.isa
1671 InstFormats:
```

```
1672 - InstFormatName: TEST69.if
1673   ISA: TEST69.isa
1674   FormatWidth: 32
1675   Fields:
1676     - FieldName: opcode
1677       FieldType: CGInstCode
1678       FieldWidth: 8
1679       StartBit: 0
1680       EndBit: 7
1681       MandatoryField: true
1682     - FieldName: RB
1683       FieldType: CGInstReg
1684       FieldWidth: 8
1685       StartBit: 8
1686       EndBit: 15
1687       MandatoryField: false
1688       RegClass: TEST69.regclass
1689     - FieldName: RA
1690       FieldType: CGInstReg
1691       FieldWidth: 8
1692       StartBit: 16
1693       EndBit: 23
1694       MandatoryField: false
1695       RegClass: TEST69.regclass
1696     - FieldName: RT
1697       FieldType: CGInstReg
1698       FieldWidth: 8
1699       StartBit: 24
1700       EndBit: 31
1701       MandatoryField: false
1702       RegClass: TEST69.csrregclass
1703 Insts:
1704 - Inst: TEST69.inst0
1705   ISA: TEST69.isa
1706   InstFormat: TEST69.if
1707   Encodings:
1708     - EncodingField: opcode
1709       EncodingWidth: 8
1710       EncodingValue: 1
1711 - Inst: TEST69.inst1
1712   ISA: TEST69.isa
1713   InstFormat: TEST69.if
1714   Encodings:
1715     - EncodingField: opcode
1716       EncodingWidth: 8
1717       EncodingValue: 2
1718 - Inst: TEST69.inst2
1719   ISA: TEST69.isa
1720   InstFormat: TEST69.if
1721   Encodings:
1722     - EncodingField: opcode
1723       EncodingWidth: 8
1724       EncodingValue: 3
1725 - Inst: TEST69.inst3
```

```
1726     ISA: TEST69.isa
1727     InstFormat: TEST69.if
1728     Encodings:
1729         - EncodingField: opcode
1730           EncodingWidth: 8
1731           EncodingValue: 4
1732 - Inst: TEST69.inst4
1733     ISA: TEST69.isa
1734     InstFormat: TEST69.if
1735     Encodings:
1736         - EncodingField: opcode
1737           EncodingWidth: 8
1738           EncodingValue: 5
1739 - Inst: TEST69.inst5
1740     ISA: TEST69.isa
1741     InstFormat: TEST69.if
1742     Encodings:
1743         - EncodingField: opcode
1744           EncodingWidth: 8
1745           EncodingValue: 6
1746 - Inst: TEST69.inst6
1747     ISA: TEST69.isa
1748     InstFormat: TEST69.if
1749     Encodings:
1750         - EncodingField: opcode
1751           EncodingWidth: 8
1752           EncodingValue: 7
1753 - Inst: TEST69.inst7
1754     ISA: TEST69.isa
1755     InstFormat: TEST69.if
1756     Encodings:
1757         - EncodingField: opcode
1758           EncodingWidth: 8
1759           EncodingValue: 8
1760 - Inst: TEST69.inst8
1761     ISA: TEST69.isa
1762     InstFormat: TEST69.if
1763     Encodings:
1764         - EncodingField: opcode
1765           EncodingWidth: 8
1766           EncodingValue: 9
1767 - Inst: TEST69.inst9
1768     ISA: TEST69.isa
1769     InstFormat: TEST69.if
1770     Encodings:
1771         - EncodingField: opcode
1772           EncodingWidth: 8
1773           EncodingValue: 10
1774 - Inst: TEST69.inst10
1775     ISA: TEST69.isa
1776     InstFormat: TEST69.if
1777     Encodings:
1778         - EncodingField: opcode
1779           EncodingWidth: 8
```

```
1780     EncodingValue: 11
1781 - Inst: TEST69.inst11
1782   ISA: TEST69.isa
1783   InstFormat: TEST69.if
1784   Encodings:
1785     - EncodingField: opcode
1786       EncodingWidth: 8
1787       EncodingValue: 12
1788 - Inst: TEST69.inst12
1789   ISA: TEST69.isa
1790   InstFormat: TEST69.if
1791   Encodings:
1792     - EncodingField: opcode
1793       EncodingWidth: 8
1794       EncodingValue: 13
1795 - Inst: TEST69.inst13
1796   ISA: TEST69.isa
1797   InstFormat: TEST69.if
1798   Encodings:
1799     - EncodingField: opcode
1800       EncodingWidth: 8
1801       EncodingValue: 14
1802 - Inst: TEST69.inst14
1803   ISA: TEST69.isa
1804   InstFormat: TEST69.if
1805   Encodings:
1806     - EncodingField: opcode
1807       EncodingWidth: 8
1808       EncodingValue: 15
1809 - Inst: TEST69.inst15
1810   ISA: TEST69.isa
1811   InstFormat: TEST69.if
1812   Encodings:
1813     - EncodingField: opcode
1814       EncodingWidth: 8
1815       EncodingValue: 16
1816 - Inst: TEST69.inst16
1817   ISA: TEST69.isa
1818   InstFormat: TEST69.if
1819   Encodings:
1820     - EncodingField: opcode
1821       EncodingWidth: 8
1822       EncodingValue: 17
1823 - Inst: TEST69.inst17
1824   ISA: TEST69.isa
1825   InstFormat: TEST69.if
1826   Encodings:
1827     - EncodingField: opcode
1828       EncodingWidth: 8
1829       EncodingValue: 18
1830 - Inst: TEST69.inst18
1831   ISA: TEST69.isa
1832   InstFormat: TEST69.if
1833   Encodings:
```

```
1834     - EncodingField: opcode
1835       EncodingWidth: 8
1836       EncodingValue: 19
1837 - Inst: TEST69.inst19
1838   ISA: TEST69.isa
1839   InstFormat: TEST69.if
1840   Encodings:
1841     - EncodingField: opcode
1842       EncodingWidth: 8
1843       EncodingValue: 20
1844 - Inst: TEST69.inst20
1845   ISA: TEST69.isa
1846   InstFormat: TEST69.if
1847   Encodings:
1848     - EncodingField: opcode
1849       EncodingWidth: 8
1850       EncodingValue: 21
1851 - Inst: TEST69.inst21
1852   ISA: TEST69.isa
1853   InstFormat: TEST69.if
1854   Encodings:
1855     - EncodingField: opcode
1856       EncodingWidth: 8
1857       EncodingValue: 22
1858 - Inst: TEST69.inst22
1859   ISA: TEST69.isa
1860   InstFormat: TEST69.if
1861   Encodings:
1862     - EncodingField: opcode
1863       EncodingWidth: 8
1864       EncodingValue: 23
1865 - Inst: TEST69.inst23
1866   ISA: TEST69.isa
1867   InstFormat: TEST69.if
1868   Encodings:
1869     - EncodingField: opcode
1870       EncodingWidth: 8
1871       EncodingValue: 24
1872 - Inst: TEST69.inst24
1873   ISA: TEST69.isa
1874   InstFormat: TEST69.if
1875   Encodings:
1876     - EncodingField: opcode
1877       EncodingWidth: 8
1878       EncodingValue: 25
1879 - Inst: TEST69.inst25
1880   ISA: TEST69.isa
1881   InstFormat: TEST69.if
1882   Encodings:
1883     - EncodingField: opcode
1884       EncodingWidth: 8
1885       EncodingValue: 26
1886 - Inst: TEST69.inst26
1887   ISA: TEST69.isa
```



```
1888     InstFormat: TEST69.if
1889     Encodings:
1890         - EncodingField: opcode
1891           EncodingWidth: 8
1892           EncodingValue: 27
1893 - Inst: TEST69.inst27
1894   ISA: TEST69.isa
1895   InstFormat: TEST69.if
1896   Encodings:
1897       - EncodingField: opcode
1898         EncodingWidth: 8
1899         EncodingValue: 28
1900 - Inst: TEST69.inst28
1901   ISA: TEST69.isa
1902   InstFormat: TEST69.if
1903   Encodings:
1904       - EncodingField: opcode
1905         EncodingWidth: 8
1906         EncodingValue: 29
1907 - Inst: TEST69.inst29
1908   ISA: TEST69.isa
1909   InstFormat: TEST69.if
1910   Encodings:
1911       - EncodingField: opcode
1912         EncodingWidth: 8
1913         EncodingValue: 30
1914 - Inst: TEST69.inst30
1915   ISA: TEST69.isa
1916   InstFormat: TEST69.if
1917   Encodings:
1918       - EncodingField: opcode
1919         EncodingWidth: 8
1920         EncodingValue: 31
1921 - Inst: TEST69.inst31
1922   ISA: TEST69.isa
1923   InstFormat: TEST69.if
1924   Encodings:
1925       - EncodingField: opcode
1926         EncodingWidth: 8
1927         EncodingValue: 32
1928 - Inst: TEST69.inst32
1929   ISA: TEST69.isa
1930   InstFormat: TEST69.if
1931   Encodings:
1932       - EncodingField: opcode
1933         EncodingWidth: 8
1934         EncodingValue: 33
1935 - Inst: TEST69.inst33
1936   ISA: TEST69.isa
1937   InstFormat: TEST69.if
1938   Encodings:
1939       - EncodingField: opcode
1940         EncodingWidth: 8
1941         EncodingValue: 34
```

```
1942 - Inst: TEST69.inst34
1943   ISA: TEST69.isa
1944   InstFormat: TEST69.if
1945   Encodings:
1946     - EncodingField: opcode
1947       EncodingWidth: 8
1948       EncodingValue: 35
1949 - Inst: TEST69.inst35
1950   ISA: TEST69.isa
1951   InstFormat: TEST69.if
1952   Encodings:
1953     - EncodingField: opcode
1954       EncodingWidth: 8
1955       EncodingValue: 36
1956 - Inst: TEST69.inst36
1957   ISA: TEST69.isa
1958   InstFormat: TEST69.if
1959   Encodings:
1960     - EncodingField: opcode
1961       EncodingWidth: 8
1962       EncodingValue: 37
1963 - Inst: TEST69.inst37
1964   ISA: TEST69.isa
1965   InstFormat: TEST69.if
1966   Encodings:
1967     - EncodingField: opcode
1968       EncodingWidth: 8
1969       EncodingValue: 38
1970 - Inst: TEST69.inst38
1971   ISA: TEST69.isa
1972   InstFormat: TEST69.if
1973   Encodings:
1974     - EncodingField: opcode
1975       EncodingWidth: 8
1976       EncodingValue: 39
1977 - Inst: TEST69.inst39
1978   ISA: TEST69.isa
1979   InstFormat: TEST69.if
1980   Encodings:
1981     - EncodingField: opcode
1982       EncodingWidth: 8
1983       EncodingValue: 40
1984 - Inst: TEST69.inst40
1985   ISA: TEST69.isa
1986   InstFormat: TEST69.if
1987   Encodings:
1988     - EncodingField: opcode
1989       EncodingWidth: 8
1990       EncodingValue: 41
1991 - Inst: TEST69.inst41
1992   ISA: TEST69.isa
1993   InstFormat: TEST69.if
1994   Encodings:
1995     - EncodingField: opcode
```

```
1996     EncodingWidth: 8
1997     EncodingValue: 42
1998 - Inst: TEST69.inst42
1999     ISA: TEST69.isa
2000     InstFormat: TEST69.if
2001     Encodings:
2002     - EncodingField: opcode
2003     EncodingWidth: 8
2004     EncodingValue: 43
2005 - Inst: TEST69.inst43
2006     ISA: TEST69.isa
2007     InstFormat: TEST69.if
2008     Encodings:
2009     - EncodingField: opcode
2010     EncodingWidth: 8
2011     EncodingValue: 44
2012 - Inst: TEST69.inst44
2013     ISA: TEST69.isa
2014     InstFormat: TEST69.if
2015     Encodings:
2016     - EncodingField: opcode
2017     EncodingWidth: 8
2018     EncodingValue: 45
2019 - Inst: TEST69.inst45
2020     ISA: TEST69.isa
2021     InstFormat: TEST69.if
2022     Encodings:
2023     - EncodingField: opcode
2024     EncodingWidth: 8
2025     EncodingValue: 46
2026 - Inst: TEST69.inst46
2027     ISA: TEST69.isa
2028     InstFormat: TEST69.if
2029     Encodings:
2030     - EncodingField: opcode
2031     EncodingWidth: 8
2032     EncodingValue: 47
2033 - Inst: TEST69.inst47
2034     ISA: TEST69.isa
2035     InstFormat: TEST69.if
2036     Encodings:
2037     - EncodingField: opcode
2038     EncodingWidth: 8
2039     EncodingValue: 48
2040 - Inst: TEST69.inst48
2041     ISA: TEST69.isa
2042     InstFormat: TEST69.if
2043     Encodings:
2044     - EncodingField: opcode
2045     EncodingWidth: 8
2046     EncodingValue: 49
2047 - Inst: TEST69.inst49
2048     ISA: TEST69.isa
2049     InstFormat: TEST69.if
```

```
2050 Encodings:
2051   - EncodingField: opcode
2052     EncodingWidth: 8
2053     EncodingValue: 50
2054 - Inst: TEST69.inst50
2055   ISA: TEST69.isa
2056   InstFormat: TEST69.if
2057   Encodings:
2058     - EncodingField: opcode
2059       EncodingWidth: 8
2060       EncodingValue: 51
2061 - Inst: TEST69.inst51
2062   ISA: TEST69.isa
2063   InstFormat: TEST69.if
2064   Encodings:
2065     - EncodingField: opcode
2066       EncodingWidth: 8
2067       EncodingValue: 52
2068 - Inst: TEST69.inst52
2069   ISA: TEST69.isa
2070   InstFormat: TEST69.if
2071   Encodings:
2072     - EncodingField: opcode
2073       EncodingWidth: 8
2074       EncodingValue: 53
2075 - Inst: TEST69.inst53
2076   ISA: TEST69.isa
2077   InstFormat: TEST69.if
2078   Encodings:
2079     - EncodingField: opcode
2080       EncodingWidth: 8
2081       EncodingValue: 54
2082 - Inst: TEST69.inst54
2083   ISA: TEST69.isa
2084   InstFormat: TEST69.if
2085   Encodings:
2086     - EncodingField: opcode
2087       EncodingWidth: 8
2088       EncodingValue: 55
2089 - Inst: TEST69.inst55
2090   ISA: TEST69.isa
2091   InstFormat: TEST69.if
2092   Encodings:
2093     - EncodingField: opcode
2094       EncodingWidth: 8
2095       EncodingValue: 56
2096 - Inst: TEST69.inst56
2097   ISA: TEST69.isa
2098   InstFormat: TEST69.if
2099   Encodings:
2100     - EncodingField: opcode
2101       EncodingWidth: 8
2102       EncodingValue: 57
2103 - Inst: TEST69.inst57
```

```
2104   ISA: TEST69.isa
2105   InstFormat: TEST69.if
2106   Encodings:
2107     - EncodingField: opcode
2108       EncodingWidth: 8
2109       EncodingValue: 58
2110 - Inst: TEST69.inst58
2111   ISA: TEST69.isa
2112   InstFormat: TEST69.if
2113   Encodings:
2114     - EncodingField: opcode
2115       EncodingWidth: 8
2116       EncodingValue: 59
2117 - Inst: TEST69.inst59
2118   ISA: TEST69.isa
2119   InstFormat: TEST69.if
2120   Encodings:
2121     - EncodingField: opcode
2122       EncodingWidth: 8
2123       EncodingValue: 60
2124 - Inst: TEST69.inst60
2125   ISA: TEST69.isa
2126   InstFormat: TEST69.if
2127   Encodings:
2128     - EncodingField: opcode
2129       EncodingWidth: 8
2130       EncodingValue: 61
2131 - Inst: TEST69.inst61
2132   ISA: TEST69.isa
2133   InstFormat: TEST69.if
2134   Encodings:
2135     - EncodingField: opcode
2136       EncodingWidth: 8
2137       EncodingValue: 62
2138 - Inst: TEST69.inst62
2139   ISA: TEST69.isa
2140   InstFormat: TEST69.if
2141   Encodings:
2142     - EncodingField: opcode
2143       EncodingWidth: 8
2144       EncodingValue: 63
2145 - Inst: TEST69.inst63
2146   ISA: TEST69.isa
2147   InstFormat: TEST69.if
2148   Encodings:
2149     - EncodingField: opcode
2150       EncodingWidth: 8
2151       EncodingValue: 64
2152 - Inst: TEST69.inst64
2153   ISA: TEST69.isa
2154   InstFormat: TEST69.if
2155   Encodings:
2156     - EncodingField: opcode
2157       EncodingWidth: 8
```

```
2158     EncodingValue: 65
2159 - Inst: TEST69.inst65
2160   ISA: TEST69.isa
2161   InstFormat: TEST69.if
2162   Encodings:
2163     - EncodingField: opcode
2164       EncodingWidth: 8
2165       EncodingValue: 66
2166 - Inst: TEST69.inst66
2167   ISA: TEST69.isa
2168   InstFormat: TEST69.if
2169   Encodings:
2170     - EncodingField: opcode
2171       EncodingWidth: 8
2172       EncodingValue: 67
2173 - Inst: TEST69.inst67
2174   ISA: TEST69.isa
2175   InstFormat: TEST69.if
2176   Encodings:
2177     - EncodingField: opcode
2178       EncodingWidth: 8
2179       EncodingValue: 68
2180 - Inst: TEST69.inst68
2181   ISA: TEST69.isa
2182   InstFormat: TEST69.if
2183   Encodings:
2184     - EncodingField: opcode
2185       EncodingWidth: 8
2186       EncodingValue: 69
2187 PseudoInsts:
2188 - PseudoInst: TEST69.pinst0
2189   ISA: TEST69.isa
2190   Inst: TEST69.inst0
2191   Encodings:
2192     - EncodingField: RA
2193       EncodingWidth: 8
2194       EncodingValue: 0
2195 - PseudoInst: TEST69.pinst1
2196   ISA: TEST69.isa
2197   Inst: TEST69.inst1
2198   Encodings:
2199     - EncodingField: RA
2200       EncodingWidth: 8
2201       EncodingValue: 1
2202 - PseudoInst: TEST69.pinst2
2203   ISA: TEST69.isa
2204   Inst: TEST69.inst2
2205   Encodings:
2206     - EncodingField: RA
2207       EncodingWidth: 8
2208       EncodingValue: 2
2209 - PseudoInst: TEST69.pinst3
2210   ISA: TEST69.isa
2211   Inst: TEST69.inst3
```

```
2212 Encodings:
2213   - EncodingField: RA
2214     EncodingWidth: 8
2215     EncodingValue: 3
2216 - PseudoInst: TEST69.pinst4
2217   ISA: TEST69.isa
2218   Inst: TEST69.inst4
2219   Encodings:
2220     - EncodingField: RA
2221       EncodingWidth: 8
2222       EncodingValue: 4
2223 - PseudoInst: TEST69.pinst5
2224   ISA: TEST69.isa
2225   Inst: TEST69.inst5
2226   Encodings:
2227     - EncodingField: RA
2228       EncodingWidth: 8
2229       EncodingValue: 5
2230 - PseudoInst: TEST69.pinst6
2231   ISA: TEST69.isa
2232   Inst: TEST69.inst6
2233   Encodings:
2234     - EncodingField: RA
2235       EncodingWidth: 8
2236       EncodingValue: 6
2237 - PseudoInst: TEST69.pinst7
2238   ISA: TEST69.isa
2239   Inst: TEST69.inst7
2240   Encodings:
2241     - EncodingField: RA
2242       EncodingWidth: 8
2243       EncodingValue: 7
2244 - PseudoInst: TEST69.pinst8
2245   ISA: TEST69.isa
2246   Inst: TEST69.inst8
2247   Encodings:
2248     - EncodingField: RA
2249       EncodingWidth: 8
2250       EncodingValue: 8
2251 - PseudoInst: TEST69.pinst9
2252   ISA: TEST69.isa
2253   Inst: TEST69.inst9
2254   Encodings:
2255     - EncodingField: RA
2256       EncodingWidth: 8
2257       EncodingValue: 9
2258 - PseudoInst: TEST69.pinst10
2259   ISA: TEST69.isa
2260   Inst: TEST69.inst10
2261   Encodings:
2262     - EncodingField: RA
2263       EncodingWidth: 8
2264       EncodingValue: 10
2265 - PseudoInst: TEST69.pinst11
```

```
2266   ISA: TEST69.isa
2267   Inst: TEST69.inst11
2268   Encodings:
2269     - EncodingField: RA
2270       EncodingWidth: 8
2271       EncodingValue: 11
2272 - PseudoInst: TEST69.pinst12
2273   ISA: TEST69.isa
2274   Inst: TEST69.inst12
2275   Encodings:
2276     - EncodingField: RA
2277       EncodingWidth: 8
2278       EncodingValue: 12
2279 - PseudoInst: TEST69.pinst13
2280   ISA: TEST69.isa
2281   Inst: TEST69.inst13
2282   Encodings:
2283     - EncodingField: RA
2284       EncodingWidth: 8
2285       EncodingValue: 13
2286 - PseudoInst: TEST69.pinst14
2287   ISA: TEST69.isa
2288   Inst: TEST69.inst14
2289   Encodings:
2290     - EncodingField: RA
2291       EncodingWidth: 8
2292       EncodingValue: 14
2293 - PseudoInst: TEST69.pinst15
2294   ISA: TEST69.isa
2295   Inst: TEST69.inst15
2296   Encodings:
2297     - EncodingField: RA
2298       EncodingWidth: 8
2299       EncodingValue: 15
2300 - PseudoInst: TEST69.pinst16
2301   ISA: TEST69.isa
2302   Inst: TEST69.inst16
2303   Encodings:
2304     - EncodingField: RA
2305       EncodingWidth: 8
2306       EncodingValue: 16
2307 - PseudoInst: TEST69.pinst17
2308   ISA: TEST69.isa
2309   Inst: TEST69.inst17
2310   Encodings:
2311     - EncodingField: RA
2312       EncodingWidth: 8
2313       EncodingValue: 17
2314 - PseudoInst: TEST69.pinst18
2315   ISA: TEST69.isa
2316   Inst: TEST69.inst18
2317   Encodings:
2318     - EncodingField: RA
2319       EncodingWidth: 8
```



```
2320     EncodingValue: 18
2321 - PseudoInst: TEST69.pinst19
2322   ISA: TEST69.isa
2323   Inst: TEST69.inst19
2324   Encodings:
2325     - EncodingField: RA
2326       EncodingWidth: 8
2327       EncodingValue: 19
2328 - PseudoInst: TEST69.pinst20
2329   ISA: TEST69.isa
2330   Inst: TEST69.inst20
2331   Encodings:
2332     - EncodingField: RA
2333       EncodingWidth: 8
2334       EncodingValue: 20
2335 - PseudoInst: TEST69.pinst21
2336   ISA: TEST69.isa
2337   Inst: TEST69.inst21
2338   Encodings:
2339     - EncodingField: RA
2340       EncodingWidth: 8
2341       EncodingValue: 21
2342 - PseudoInst: TEST69.pinst22
2343   ISA: TEST69.isa
2344   Inst: TEST69.inst22
2345   Encodings:
2346     - EncodingField: RA
2347       EncodingWidth: 8
2348       EncodingValue: 22
2349 - PseudoInst: TEST69.pinst23
2350   ISA: TEST69.isa
2351   Inst: TEST69.inst23
2352   Encodings:
2353     - EncodingField: RA
2354       EncodingWidth: 8
2355       EncodingValue: 23
2356 - PseudoInst: TEST69.pinst24
2357   ISA: TEST69.isa
2358   Inst: TEST69.inst24
2359   Encodings:
2360     - EncodingField: RA
2361       EncodingWidth: 8
2362       EncodingValue: 24
2363 - PseudoInst: TEST69.pinst25
2364   ISA: TEST69.isa
2365   Inst: TEST69.inst25
2366   Encodings:
2367     - EncodingField: RA
2368       EncodingWidth: 8
2369       EncodingValue: 25
2370 - PseudoInst: TEST69.pinst26
2371   ISA: TEST69.isa
2372   Inst: TEST69.inst26
2373   Encodings:
```

```
2374     - EncodingField: RA
2375       EncodingWidth: 8
2376       EncodingValue: 26
2377 - PseudoInst: TEST69.pinst27
2378   ISA: TEST69.isa
2379   Inst: TEST69.inst27
2380   Encodings:
2381     - EncodingField: RA
2382       EncodingWidth: 8
2383       EncodingValue: 27
2384 - PseudoInst: TEST69.pinst28
2385   ISA: TEST69.isa
2386   Inst: TEST69.inst28
2387   Encodings:
2388     - EncodingField: RA
2389       EncodingWidth: 8
2390       EncodingValue: 28
2391 - PseudoInst: TEST69.pinst29
2392   ISA: TEST69.isa
2393   Inst: TEST69.inst29
2394   Encodings:
2395     - EncodingField: RA
2396       EncodingWidth: 8
2397       EncodingValue: 29
2398 - PseudoInst: TEST69.pinst30
2399   ISA: TEST69.isa
2400   Inst: TEST69.inst30
2401   Encodings:
2402     - EncodingField: RA
2403       EncodingWidth: 8
2404       EncodingValue: 30
2405 - PseudoInst: TEST69.pinst31
2406   ISA: TEST69.isa
2407   Inst: TEST69.inst31
2408   Encodings:
2409     - EncodingField: RA
2410       EncodingWidth: 8
2411       EncodingValue: 31
2412 - PseudoInst: TEST69.pinst32
2413   ISA: TEST69.isa
2414   Inst: TEST69.inst32
2415   Encodings:
2416     - EncodingField: RA
2417       EncodingWidth: 8
2418       EncodingValue: 32
2419 - PseudoInst: TEST69.pinst33
2420   ISA: TEST69.isa
2421   Inst: TEST69.inst33
2422   Encodings:
2423     - EncodingField: RA
2424       EncodingWidth: 8
2425       EncodingValue: 33
2426 - PseudoInst: TEST69.pinst34
2427   ISA: TEST69.isa
```

```
2428     Inst: TEST69.inst34
2429     Encodings:
2430         - EncodingField: RA
2431           EncodingWidth: 8
2432           EncodingValue: 34
2433     - PseudoInst: TEST69.pinst35
2434       ISA: TEST69.isa
2435       Inst: TEST69.inst35
2436       Encodings:
2437         - EncodingField: RA
2438           EncodingWidth: 8
2439           EncodingValue: 35
2440     - PseudoInst: TEST69.pinst36
2441       ISA: TEST69.isa
2442       Inst: TEST69.inst36
2443       Encodings:
2444         - EncodingField: RA
2445           EncodingWidth: 8
2446           EncodingValue: 36
2447     - PseudoInst: TEST69.pinst37
2448       ISA: TEST69.isa
2449       Inst: TEST69.inst37
2450       Encodings:
2451         - EncodingField: RA
2452           EncodingWidth: 8
2453           EncodingValue: 37
2454     - PseudoInst: TEST69.pinst38
2455       ISA: TEST69.isa
2456       Inst: TEST69.inst38
2457       Encodings:
2458         - EncodingField: RA
2459           EncodingWidth: 8
2460           EncodingValue: 38
2461     - PseudoInst: TEST69.pinst39
2462       ISA: TEST69.isa
2463       Inst: TEST69.inst39
2464       Encodings:
2465         - EncodingField: RA
2466           EncodingWidth: 8
2467           EncodingValue: 39
2468     - PseudoInst: TEST69.pinst40
2469       ISA: TEST69.isa
2470       Inst: TEST69.inst40
2471       Encodings:
2472         - EncodingField: RA
2473           EncodingWidth: 8
2474           EncodingValue: 40
2475     - PseudoInst: TEST69.pinst41
2476       ISA: TEST69.isa
2477       Inst: TEST69.inst41
2478       Encodings:
2479         - EncodingField: RA
2480           EncodingWidth: 8
2481           EncodingValue: 41
```

```
2482 - PseudoInst: TEST69.pinst42
2483   ISA: TEST69.isa
2484   Inst: TEST69.inst42
2485   Encodings:
2486     - EncodingField: RA
2487       EncodingWidth: 8
2488       EncodingValue: 42
2489 - PseudoInst: TEST69.pinst43
2490   ISA: TEST69.isa
2491   Inst: TEST69.inst43
2492   Encodings:
2493     - EncodingField: RA
2494       EncodingWidth: 8
2495       EncodingValue: 43
2496 - PseudoInst: TEST69.pinst44
2497   ISA: TEST69.isa
2498   Inst: TEST69.inst44
2499   Encodings:
2500     - EncodingField: RA
2501       EncodingWidth: 8
2502       EncodingValue: 44
2503 - PseudoInst: TEST69.pinst45
2504   ISA: TEST69.isa
2505   Inst: TEST69.inst45
2506   Encodings:
2507     - EncodingField: RA
2508       EncodingWidth: 8
2509       EncodingValue: 45
2510 - PseudoInst: TEST69.pinst46
2511   ISA: TEST69.isa
2512   Inst: TEST69.inst46
2513   Encodings:
2514     - EncodingField: RA
2515       EncodingWidth: 8
2516       EncodingValue: 46
2517 - PseudoInst: TEST69.pinst47
2518   ISA: TEST69.isa
2519   Inst: TEST69.inst47
2520   Encodings:
2521     - EncodingField: RA
2522       EncodingWidth: 8
2523       EncodingValue: 47
2524 - PseudoInst: TEST69.pinst48
2525   ISA: TEST69.isa
2526   Inst: TEST69.inst48
2527   Encodings:
2528     - EncodingField: RA
2529       EncodingWidth: 8
2530       EncodingValue: 48
2531 - PseudoInst: TEST69.pinst49
2532   ISA: TEST69.isa
2533   Inst: TEST69.inst49
2534   Encodings:
2535     - EncodingField: RA
```

```
2536         EncodingWidth: 8
2537         EncodingValue: 49
2538 - PseudoInst: TEST69.pinst50
2539   ISA: TEST69.isa
2540   Inst: TEST69.inst50
2541   Encodings:
2542     - EncodingField: RA
2543       EncodingWidth: 8
2544       EncodingValue: 50
2545 - PseudoInst: TEST69.pinst51
2546   ISA: TEST69.isa
2547   Inst: TEST69.inst51
2548   Encodings:
2549     - EncodingField: RA
2550       EncodingWidth: 8
2551       EncodingValue: 51
2552 - PseudoInst: TEST69.pinst52
2553   ISA: TEST69.isa
2554   Inst: TEST69.inst52
2555   Encodings:
2556     - EncodingField: RA
2557       EncodingWidth: 8
2558       EncodingValue: 52
2559 - PseudoInst: TEST69.pinst53
2560   ISA: TEST69.isa
2561   Inst: TEST69.inst53
2562   Encodings:
2563     - EncodingField: RA
2564       EncodingWidth: 8
2565       EncodingValue: 53
2566 - PseudoInst: TEST69.pinst54
2567   ISA: TEST69.isa
2568   Inst: TEST69.inst54
2569   Encodings:
2570     - EncodingField: RA
2571       EncodingWidth: 8
2572       EncodingValue: 54
2573 - PseudoInst: TEST69.pinst55
2574   ISA: TEST69.isa
2575   Inst: TEST69.inst55
2576   Encodings:
2577     - EncodingField: RA
2578       EncodingWidth: 8
2579       EncodingValue: 55
2580 - PseudoInst: TEST69.pinst56
2581   ISA: TEST69.isa
2582   Inst: TEST69.inst56
2583   Encodings:
2584     - EncodingField: RA
2585       EncodingWidth: 8
2586       EncodingValue: 56
2587 - PseudoInst: TEST69.pinst57
2588   ISA: TEST69.isa
2589   Inst: TEST69.inst57
```

```
2590 Encodings:
2591   - EncodingField: RA
2592     EncodingWidth: 8
2593     EncodingValue: 57
2594 - PseudoInst: TEST69.pinst58
2595   ISA: TEST69.isa
2596   Inst: TEST69.inst58
2597   Encodings:
2598     - EncodingField: RA
2599       EncodingWidth: 8
2600       EncodingValue: 58
2601 - PseudoInst: TEST69.pinst59
2602   ISA: TEST69.isa
2603   Inst: TEST69.inst59
2604   Encodings:
2605     - EncodingField: RA
2606       EncodingWidth: 8
2607       EncodingValue: 59
2608 - PseudoInst: TEST69.pinst60
2609   ISA: TEST69.isa
2610   Inst: TEST69.inst60
2611   Encodings:
2612     - EncodingField: RA
2613       EncodingWidth: 8
2614       EncodingValue: 60
2615 - PseudoInst: TEST69.pinst61
2616   ISA: TEST69.isa
2617   Inst: TEST69.inst61
2618   Encodings:
2619     - EncodingField: RA
2620       EncodingWidth: 8
2621       EncodingValue: 61
2622 - PseudoInst: TEST69.pinst62
2623   ISA: TEST69.isa
2624   Inst: TEST69.inst62
2625   Encodings:
2626     - EncodingField: RA
2627       EncodingWidth: 8
2628       EncodingValue: 62
2629 - PseudoInst: TEST69.pinst63
2630   ISA: TEST69.isa
2631   Inst: TEST69.inst63
2632   Encodings:
2633     - EncodingField: RA
2634       EncodingWidth: 8
2635       EncodingValue: 63
2636 - PseudoInst: TEST69.pinst64
2637   ISA: TEST69.isa
2638   Inst: TEST69.inst64
2639   Encodings:
2640     - EncodingField: RA
2641       EncodingWidth: 8
2642       EncodingValue: 64
2643 - PseudoInst: TEST69.pinst65
```

```
2644     ISA: TEST69.isa
2645     Inst: TEST69.inst65
2646     Encodings:
2647         - EncodingField: RA
2648           EncodingWidth: 8
2649           EncodingValue: 65
2650     - PseudoInst: TEST69.pinst66
2651       ISA: TEST69.isa
2652       Inst: TEST69.inst66
2653       Encodings:
2654           - EncodingField: RA
2655             EncodingWidth: 8
2656             EncodingValue: 66
2657     - PseudoInst: TEST69.pinst67
2658       ISA: TEST69.isa
2659       Inst: TEST69.inst67
2660       Encodings:
2661           - EncodingField: RA
2662             EncodingWidth: 8
2663             EncodingValue: 67
2664     - PseudoInst: TEST69.pinst68
2665       ISA: TEST69.isa
2666       Inst: TEST69.inst68
2667       Encodings:
2668           - EncodingField: RA
2669             EncodingWidth: 8
2670             EncodingValue: 68
2671     Caches:
2672         - Cache: TEST69.L2.cache
2673           Sets: 2
2674           Ways: 8
2675         - Cache: TEST69.core0.L1.cache
2676           Sets: 1
2677           Ways: 1
2678           SubLevel: TEST69.L2.cache
2679         - Cache: TEST69.L2.cache
2680           Sets: 2
2681           Ways: 8
2682         - Cache: TEST69.core1.L1.cache
2683           Sets: 1
2684           Ways: 1
2685           SubLevel: TEST69.L2.cache
2686         - Cache: TEST69.L2.cache
2687           Sets: 2
2688           Ways: 8
2689         - Cache: TEST69.core2.L1.cache
2690           Sets: 1
2691           Ways: 1
2692           SubLevel: TEST69.L2.cache
2693         - Cache: TEST69.L2.cache
2694           Sets: 2
2695           Ways: 8
2696         - Cache: TEST69.core3.L1.cache
2697           Sets: 1
```

```
2698     Ways: 1
2699     SubLevel: TEST69.L2.cache
2700     - Cache: TEST69.L2.cache
2701     Sets: 2
2702     Ways: 8
2703 Comms:
2704     - Comm: TEST69.comm
2705     Type: NOC
2706     Width: 64
2707     Endpoints:
2708         - TEST69.core0
2709         - TEST69.core1
2710         - TEST69.core2
2711         - TEST69.core3
2712         - TEST69.L2.cache
2713         - TEST69.vtp
2714         - TEST69.0.mctrl
2715         - TEST69.1.mctrl
2716         - TEST69.2.mctrl
2717         - TEST69.3.mctrl
2718 Scratchpads:
2719     - Scratchpad: TEST69.0.spad
2720     MemSize: 1024
2721     RqstPorts: 1
2722     RspPorts: 1
2723     StartAddr: 34359739392
2724     - Scratchpad: TEST69.1.spad
2725     MemSize: 2048
2726     RqstPorts: 1
2727     RspPorts: 1
2728     StartAddr: 34359740416
2729     - Scratchpad: TEST69.2.spad
2730     MemSize: 3072
2731     RqstPorts: 1
2732     RspPorts: 1
2733     StartAddr: 34359741440
2734     - Scratchpad: TEST69.3.spad
2735     MemSize: 4096
2736     RqstPorts: 1
2737     RspPorts: 1
2738     StartAddr: 34359742464
2739 MemoryControllers:
2740     - MemoryController: TEST69.0.mctrl
2741     Ports: 2
2742     - MemoryController: TEST69.1.mctrl
2743     Ports: 2
2744     - MemoryController: TEST69.2.mctrl
2745     Ports: 2
2746     - MemoryController: TEST69.3.mctrl
2747     Ports: 2
2748 VTPControllers:
2749     - VTP: TEST69.vtp
2750 Extensions:
2751     - Extension: TEST69.ext0
```



```
2752     Type: unknown
2753     Registers:
2754         - RegName: TEST69.ext0.reg
2755           Width: 64
2756           Index: 0
2757           IsFixedValue: false
2758           IsSIMD: false
2759           RWReg: false
2760           ROReg: false
2761           CSRReg: true
2762           AMSReg: false
2763           Shared: false
2764     RegClasses:
2765         - RegisterClassName: TEST69.ext0.regclass
2766           Registers:
2767             - TEST69.ext0.reg
2768     ISAs:
2769         - ISAName: TEST69.ext0.isa
2770     RTLFile: TEST69.ext0.v
2771 - Extension: TEST69.ext1
2772     Type: unknown
2773     Registers:
2774         - RegName: TEST69.ext1.reg
2775           Width: 64
2776           Index: 0
2777           IsFixedValue: false
2778           IsSIMD: false
2779           RWReg: false
2780           ROReg: false
2781           CSRReg: true
2782           AMSReg: false
2783           Shared: false
2784     RegClasses:
2785         - RegisterClassName: TEST69.ext1.regclass
2786           Registers:
2787             - TEST69.ext1.reg
2788     ISAs:
2789         - ISAName: TEST69.ext1.isa
2790     RTLFile: TEST69.ext1.v
2791 - Extension: TEST69.ext2
2792     Type: unknown
2793     Registers:
2794         - RegName: TEST69.ext2.reg
2795           Width: 64
2796           Index: 0
2797           IsFixedValue: false
2798           IsSIMD: false
2799           RWReg: false
2800           ROReg: false
2801           CSRReg: true
2802           AMSReg: false
2803           Shared: false
2804     RegClasses:
2805         - RegisterClassName: TEST69.ext2.regclass
```

```
2806     Registers:
2807         - TEST69.ext2.reg
2808     ISAs:
2809         - ISAName: TEST69.ext2.isa
2810     RTLFile: TEST69.ext2.v
2811 - Extension: TEST69.ext3
2812     Type: unknown
2813     Registers:
2814         - RegName: TEST69.ext3.reg
2815           Width: 64
2816           Index: 0
2817           IsFixedValue: false
2818           IsSIMD: false
2819           RWReg: false
2820           ROReg: false
2821           CSRReg: true
2822           AMSReg: false
2823           Shared: false
2824     RegClasses:
2825         - RegisterClassName: TEST69.ext3.regclass
2826     Registers:
2827         - TEST69.ext3.reg
2828     ISAs:
2829         - ISAName: TEST69.ext3.isa
2830     RTLFile: TEST69.ext3.v
2831 Cores:
2832 - Core: TEST69.core0
2833   Cache: TEST69.core0.L1.cache
2834   ISA: TEST69.isa
2835   RegisterClasses:
2836     - RegClass: TEST69.regclass
2837     - RegClass: TEST69.csrregclass
2838   Extensions:
2839     - Extension: TEST69.ext0
2840 - Core: TEST69.core1
2841   Cache: TEST69.core1.L1.cache
2842   ISA: TEST69.isa
2843   RegisterClasses:
2844     - RegClass: TEST69.regclass
2845     - RegClass: TEST69.csrregclass
2846   Extensions:
2847     - Extension: TEST69.ext1
2848 - Core: TEST69.core2
2849   Cache: TEST69.core2.L1.cache
2850   ISA: TEST69.isa
2851   RegisterClasses:
2852     - RegClass: TEST69.regclass
2853     - RegClass: TEST69.csrregclass
2854   Extensions:
2855     - Extension: TEST69.ext2
2856 - Core: TEST69.core3
2857   Cache: TEST69.core3.L1.cache
2858   ISA: TEST69.isa
2859   RegisterClasses:
```

```
2860     - RegClass: TEST69.regclass
2861     - RegClass: TEST69.csrregclass
2862     Extensions:
2863     - Extension: TEST69.ext3
2864     Socs:
2865     - Soc: TEST69.soc
2866     Cores:
2867     - Core: TEST69.core0
2868     - Core: TEST69.core1
2869     - Core: TEST69.core2
2870     - Core: TEST69.core3
```

Listing 24: Sample IR File

References

- [1] O. Ben-Kiki, C. Evans, and I. dotNet, “Yaml ain’t markup language (yaml) version 1.2,” 2009. [Online]. Available: <http://yaml.org/spec/1.2/spec.html>