zolcgen user manual

Title	zolcgen (ZOLC initialization sequence generator for SALTO)
Author	Nikolaos Kavvadias 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016
Contact	nikos@nkavvadias.com
Website	http://www.nkavvadias.com
Release Date	30 April 2016
Version	0.2.0
Rev. history	
v0.2.0	2016-04-30 Moved to github; updated README to RestructuredText.
v0.1.0	2006-07-11 Initial release.

1. Introduction

`zolcgen` is a transformation pass operating on `SUIFrm` (SUIF "real" machine) assembly files, utilizing the [SALTO] (System for Assembly Language Transformation and Optimization) API.

The `zolcgen` SALTO pass produces the actual ZOLC (Zero Overhead Loop Controller) initialization code that has to be inserted in a preceeding basic block to the loop nest to update the ZOLC storage resources and is typically the first basic block of the targeted procedure.

This pass works for the `SUIFrm` instruction set and has been created for SALTO version 1.4.1beta3. Also, a SALTO distribution that has been modified (patched) in order to include support for the suifrm family backend should be used.

More on SALTO can be found in [Rohou96].

2. File listing

The `zolcgen` distribution includes the following files:

/zolcgen	Top-level directory
AUTHORS	List of `zolcgen` authors.
LICENSE	The modified BSD license governs `tcfggen`.

Makefile.suifvm	Makefile for the `zolcgen` SALTO pass for the `SUIFvm` (aka `SUIFrm` architecture) SALTO backend.
README.html	HTML version of README.rst.
README.pdf	PDF version of README.rst.
README.rst	This file.
VERSION	Current version of the project sources.
rst2docs.sh	Bash script for generating the HTML and PDF versions of the documentation (README.rst).
zolcgen.cc	This pass updates a SUIFrm assembly program with the proper ZOLC initialization sequence.

3. Technical information

The operation of `zolcgen` involves the following tasks:

1) Conversion of `LOOP` pseudos to an `LDSA-LDSI-LDSS-LDSF` sequence of instructions (and their repositioning).

The `LDS[I|S|F]` instructions are used for initializing loop bound (initial, final) and stride (step) values. An `LDSA` (Set index register alias) instruction associates the general-purpose register used for indexing of a certain loop with its loop address (`loop_a`). `LDSA` instructions can be used for static renaming in case of an architecture with dedicated loop index registers. For a homogeneous register architecture this information would be mapped on a small LUT providing the alias relation of registers used for indexing with their correspondent loop addresses.

2) The overhead instructions are properly handled and the respective overhead markers are removed.

The `state` of an overhead marker can be one of the following:

- KEEP (0): Do not alter the marked instruction. This is the default option.
- REPLACE_NOP (1): Replace the marked instruction with a NOP (no operation).
- REMOVE (2): Remove the marked instruction.

In all cases, the overhead marker is removed as well.

3) The task entry and exit PC addresses are calculated and the corresponding `LDS[N|X]` instructions (Set a PC entry/exit address) are created.

A `LOOP` provides a given loop address (`loop_a`), the actual loop index register (`rindex`) and the loop parameters (`initial`, `step`, `final`). This pseudo is attached to the last instruction in the basic block (typically a `BLT` = `branch if less than`).

An `OVERHEAD` marks its following non-pseudo instruction whether it must be kept (`state=0` which is the default), replaced by a no-operation (`state=1`), or entirely removed (`state=2`). These pseudos are attached to the specific instructions.

4. Installation

- Install SALTO (assuming version 1.4.1beta3 and a working gcc/g++ 2.96 host compiler). You will need a modified SALTO supporting the suifvm architecture family.
- Unpack the zolcgen archive wherever you like.
- Add the path to the SALTO shared libs (typically this is: $SALTO_HOME/lib) to $LD_LIBRARY_PATH.
- Type make -f Makefile.suifvm to build the zolcgen pass.

With minimal work, the zolcgen could be added to the salto-1.4.1beta3 source code in order to be built along with the SALTO infrastructure according to the typical GNU configuration/build procedure:

```
configure [options] ; make ; make install
```

If you are using a Linux Redhat 7.3 or 9.0 distribution, the following options will work:

```
$ CC=gcc296 CXX=g++296 --prefix=/path/to/salto-install \
  --enable-all-targets
```

If you are only interested in building the suifvm related libraries, the following can be used:

```
$ CC=gcc296 CXX=g++296 --prefix=/path/to/salto-install \
  --enable-suifvm
```

5. Usage details

The pass accepts an input file (*.s) in SUIFrm assembly form to operate. The output file is the transformed assembly file (*.zolc.s) which should be ready for simulation on a ZOLC-aware SUIFrm model, e.g., written for [ArchC].

Usage synopsys:

```
$ zolcgen [options] -m /path/to/suifrm-md/md-file.md \
  -i assembly.s -- -o assembly.zolc.s
```

where options can be one (or more) of the following:

-disable-ldsa Disable the translation of "ldsa" instructions.

-text-addr <value> Set the base address (in bytes) of the .text segment (instruction memory) to <value>, written in decade form. The default value is 4096 (0x1000).

-proc <opt-unit> Specify the name of the procedure to process with tcfggen.

6. Known limitations

1. ZOLC initialization instructions can only be appended to the first basic block of the procedure of interest.

7. References

[ArchC] The ArchC resource center. Available: http://www.archc.org

[Rohou96] E. Rohou, F. Bodin, A. Seznec, G. L. Fol, F. Charot, and F. Raimbault, SALTO: System for assembly-language transformation and optimization,. Institut National de Recherche en Informatique et en Automatique, Technical report 2980, September 1996.

[SALTO] The SALTO Project homepage. Available: http://www.irisa.fr/caps/projects/Salto/