Разработка генератора тестов для верификации устройства передачи управления микропроцессора R2000+

Выполнил студент группы M01-903a Самирханов Д.И.

Московский физико-технический институт
(государственный университет)
Физтех-школа радиотехники и компьютерных технологий
Кафедра информатики и вычислительной техники

2021

Введение

- Микропроцессор R2000+ реализует систему команд архитектуры SPARC v9, содержащую инструкции передачи управления.
- Инструкция, исполняющаяся перед завершением инструкции передачи управления, находится в задержанном слоте (delay-слоте).
- Инструкции перехода содержат признак annul, отменяющий исполнение инструкции в delay-слоте для:
  - безусловных переходов;
  - not-taken-ветви условных переходов.
- Система передачи управления микропроцессора R2000+ включает в себя комбинированный предсказатель переходов, использующий массив бимодальных счетчиков.

Инструкции передачи управления

Безусловная передача управления

группа инструкций	форма адресации	new PC	new nPC
BA	смещ. отн. PC	nPC	EA
BA,a		EA	EA+4
CALL		nPC	EA
JMPL / RETURN	косв. рег. адресация		

Условная передача управления

группа инструкций	форма адресации	taken	new PC	new nPC
BPr/BPcc	смещ. отн. PC	да	nPC	EA
		нет		nPC + 4
BPr,a/BPcc,a		да		EA
		нет	nPC + 4	nPC + 8

Прочие инструкции

группа инструкций	new PC	new nPC
Non-CTIs	nPC	nPC + 4

- PC – Program Counter, адрес текущей исполняемой инструкции
- nPC – next Program Counter, адрес следующей инструкции для исполнения
- EA – Effective Address, эффективный адрес
- Non-CTIs – инструкции без передачи управления
- BPr/BPcc – инструкции перехода по условию регистра и флагов соответственно

Цель работы

Разработка генератора тестов подсистемы управления микропроцессора R2000+.

Задачи:

- генерация случайного графа управления с ветвлениями и циклами;
- генерация инструкций передачи управления в delay-слотах;
- генерация случайной трассы исполнения;
- генерация трассы исполнения в режиме проверки бимодальных счетчиков;
- параметризация тестовых сценариев генерации.

Реализованный алгоритм генерации случайного теста

1. Генерация случайного графа управления.
   - Граф управления – ориентированный связный граф, вершиной которого является последовательность инструкций, завершающаяся инструкцией передачи управления. Ребра задают переходы без учета инструкций в delay-слотах. Вершины упорядочены по адресу, последняя вершина называется терминальной.
2. Генерация трассы исполнения.
   - Трасса исполнения – последовательность переходов между вершинами графа управления.
3. Генерация управляющего кода.
   - Управляющий код – код, реализующий передачу управления по графу управления согласно трассе исполнения.

Генерация случайного графа управления

- Параметры вершины графа управления:

параметр	область значений
тип вершины	{Control, Delay}
длина блока инструкций	$\mathbb{N}$ для Control, {1} для Delay
конечная инструкция	{BPr/BPcc; BA; CALL; JMPL/RETURN} для Control, {BPr/BPcc; BA; CALL; JMPL/RETURN; NonCTI} для Delay
номера вершин для переходов	[0; размер графа управления)

- Алгоритм генерации графа управления:
  1. формируется последовательность заданной длины, состоящая из вершин со случайными параметрами на основе заданных ограничений;
  2. генерируются дуги на основе инструкций и типов вершин:
     - целевые вершины задаются случайным образом;
     - для вершин с безусловными переходами: дуга на целевую вершину;
     - для вершин с условными переходами: 2 дуги – taken на целевую вершину и not taken на следующую вершину;
     - для остальных вершин: дуга на следующую вершину.

Генерация трассы исполнения

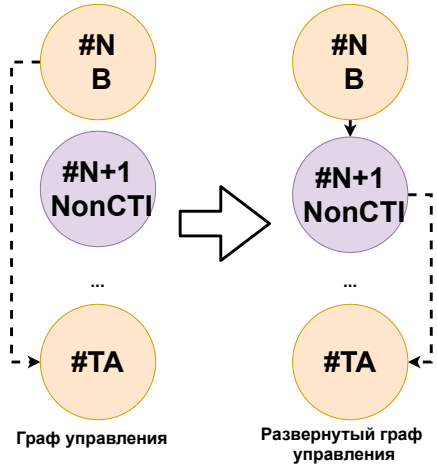
Развернутый граф управления

- Для формирования трассы исполнения необходимо учитывать инструкции в delay-слотах.
- Строится развернутый граф управления.
  Для каждой пары последовательных вершин графа управления, в зависимости от их типов, в развернутый граф управления добавляются вершины и дуги.

Генерация трассы исполнения

Формирование развернутого графа управления:: {безусловный переход; NonCTI}

B ∈ (BA, CALL, JMPL, RETURN)



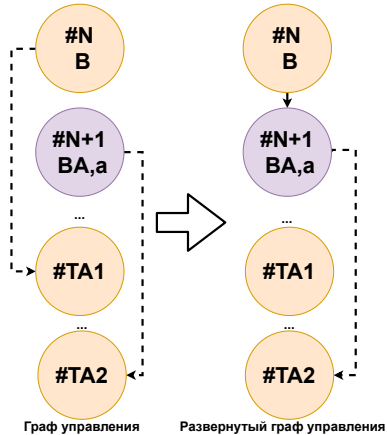
Добавляются вершины: #N, #N+1, #TA,
дуги: (#N, #N+1), (#N+1, #TA)

Инструкция в delay-слоте безусловных переходов без annul-бита всегда исполняется перед передачей управления.

Генерация трассы исполнения

Формирование развернутого графа управления:: {безусловный переход; BA,a}

B ∈ (BA, CALL, JMPL, RETURN)



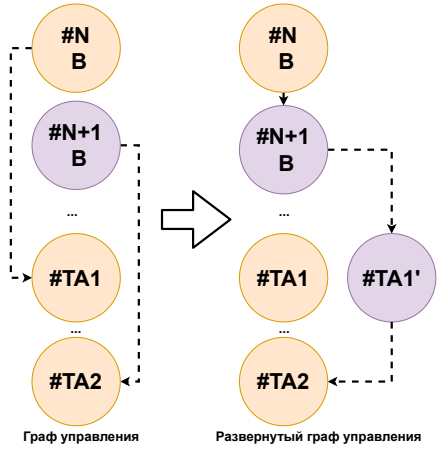
Безусловный branch с annul-битом (BA,a) в delay-слоте отменяет прыдыдущий переход.

Добавляются вершины: #N, #N+1, #TA1, #TA2,
дуги: (#N, #N+1), (#N+1, #TA2)

Генерация трассы исполнения

Формирование развернутого графа управления:: {безусловный переход; безусловный переход}

B ∈ (BA, CALL, JMPL, RETURN)



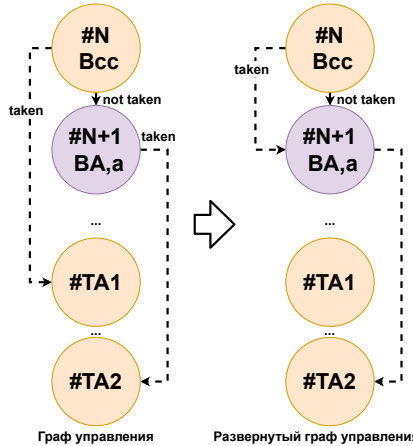
Для безусловного перехода в delay-слоте безусловного перехода происходит передача управления на #TA1. После исполнения одной инструкции по адресу #TA1 передается управление на #TA2.

Добавляются вершины: #N, #N+1, #TA1, #TA1',#TA2,
дуги: (#N, #N+1), (#N+1, #TA1'), (#TA1', #TA2)

Генерация трассы исполнения

Формирование развернутого графа управления:: {условный переход; BA,a}

Bcc ∈ (BPcc, BPr)



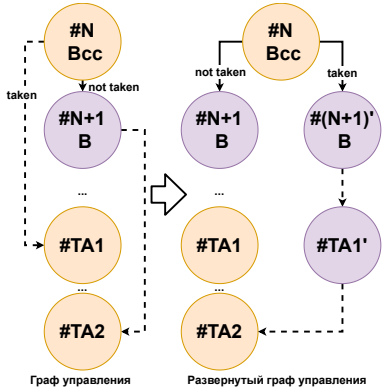
Безусловный branch с annul-битом (BA,a) в delay-слоте отменяет прыдыдущий переход.

Добавляются вершины: #N, #N+1, #TA1, #TA2,
дуги: taken(#N, #N+1), not taken(#N, #N+1), (#N+1, #TA2)

Генерация трассы исполнения

Формирование развернутого графа управления:: {условный переход; безусловный переход}

B ∈ (BA, CALL, JMPL, RETURN)
Bcc ∈ (BPcc, BPr)



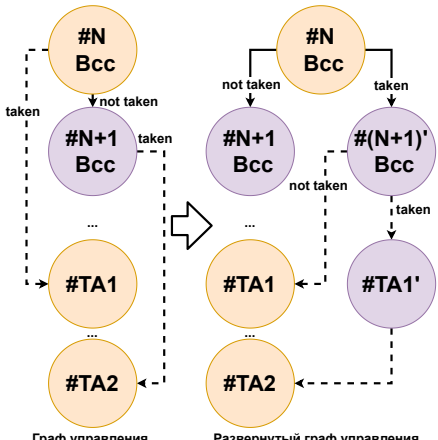
Если переход #N взят, после исполнения #N+1 передается управление на #TA1, после исполнения 1 инструкции по адресу #TA1 передается управление на #TA2.

Добавляются вершины: #N, #N+1, #(N+1)', #TA1, #TA1', #TA2,
дуги: taken(#N, #(N+1)'), not taken(#N, #N+1), (#(N+1)', #TA1'), (#TA1', #TA2)

Генерация трассы исполнения

Формирование развернутого графа управления:: {условный переход; условный переход}

Bcc ∈ (BPcc, BPr)



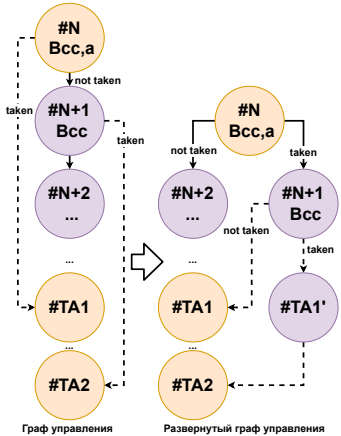
Если переход #N взят, после исполнения #N+1 передается управление на #TA1. Если #N+1 взят, то после исполнения 1 инструкции по адресу #TA1 передается управление на #TA2.

Добавляются вершины: #N, #N+1, #(N+1)', #TA1, #TA1', #TA2,
дуги: taken(#N, #(N+1)'), not taken(#N, #N+1), taken(#(N+1)', #TA1'), not taken(#(N+1)', #TA1), (#TA1', #TA2)

Генерация трассы исполнения

Формирование развернутого графа управления:: {условный переход, annul; условный переход}

Bcc ∈ (BPcc, BPr)



Если переход #N взят, после исполнения #N+1 передается управление на #TA1. Если #N+1 взят, то после исполнения 1 инструкции #TA1, передается управление на #TA2. Если переход #N не взят, игнорируется delay-слот.

Добавляются вершины: #N, #N+1, #N+2, #TA1, #TA1', #TA2,
дуги: taken(#N, #(N+1)'), not taken(#N, #N+2), taken(#N+1, #TA1'), not taken(#N+1, #TA1)

Генерация случайной трассы исполнения

Алгоритм

1. Указатель текущей исследуемой вершины указывает на нулевую вершину развернутого графа управления.
2. Пока есть переход из текущей вершины на нетерминальную и размер сгенерированной трассы меньше параметризованного размера:
   - 2.1 выбирается случайный переход на нетерминальную вершину, добавляется в трассу;
   - 2.2 обновляется указатель на текущую вершину.
3. Ищется путь из текущей вершины к терминальной, заносится в трассу.

Генерация трассы исполнения в режиме проверки бимодальных счетчиков

- Бимодальный предсказатель – массив из 4 столбцов 2-битных счетчиков высотой 256.
- Адрес ячейки бимодального массива задается битами [11:4] PC инструкции, биты [3:2] PC определяют столбец.
- При ограничении размера теста в памяти 4Кбайт не будет пересечений по ячейкам для инструкций условных переходов.
- Счетчик инкрементируется при каждом успешном предсказании и декрементируется при ошибках.
- Старший бит счетчика определяет предсказание: 1 – predict taken, 0 – predict not taken.
- Необходима генерация трассы исполнения, осуществляющей для инструкций условного перехода перебор сочетаний предсказанности и исполненности: (predict taken, taken), (predict taken, not taken), (predict not taken, taken), (predict not taken, not taken).

Генерация трассы исполнения в режиме проверки бимодальных счетчиков

Алгоритм

1. Ищется путь от начальной вершины развернутого графа управления к первой вершине с условным переходом, добавляется в трассу.
2. Для каждой вершины с условным переходом:
   - 2.1 если не первая вершина с условным переходом: ищется путь от предыдущей вершины с условным переходом к текущей, добавляется в трассу;
   - 2.2 ищется цикл с taken-переходом;
   - 2.3 ищется цикл с not-taken-переходом;
   - 2.4 последовательно в трассу добавляется 4 цикла с taken-переходом, 4 цикла с not-taken-переходом, taken-переход;
3. Ищется путь к терминальной вершине, добавляется в трассу.

Генерация управляющего кода

- Трасса исполнения определяет историю взятия условных переходов.
- В памяти теста формируется массив данных A, соответствующий трассе исполнения.
- Управляющий код по массиву A задает условие, будет ли взят переход.
- Управляющий код вставляется в линейные блоки инструкций перед условным переходом.

управляющий код условных переходов по значению регистра

1) чтение условия перехода из памяти в регистр	ldub A, R
2) инкремент указателя на память	add A, 0x1, A

управляющий код условных переходов по флагам

1) чтение условия перехода из памяти в регистр	ldub A, R
2) сравнение значения регистра с константой	subcc R, 0, %g0
3) инкремент указателя на память	add A, 0x1, A

Параметризация тестовых сценариев генерации

- Конфигурационный файл подается на входы генераторов графа и трассы для параметризации.
- В конфигурационном файле задаются параметры графа и трассы исполнения.
- Граф управления задается строкой по правилам:
  - задается набор переменных, каждая из которых соответствует единичной вершине графа управления;
  - параметры вершины задаются через описание соответствующей переменной;
  - строка содержит последовательность переменных, порядок соотвествует порядку вершин графа управления;
- Для трассы управления параметризуется режим генерации и длина трассы для режима случайной генерации.

Результаты

- Разработан генератор, включающий в себя:
  - генератор случайного графа управления с ветвлениями и циклами;
  - генератор трассы исполнения;
  - генератор управляющего кода.
- Реализована возможность параметризации тестовых сценариев генерации.
- Разработанный генератор позволил осуществить целенаправленный перебор пар предсказанности и исполненности для различных инструкций в delay-слотах.
- Генератор включен в систему автоматизированных прогонов.
- Найдено 2 ошибки в RTL-описании микропроцессора R2000+.