УДК 681.3.06

С.Ю. ГАВРИЛЕНКО, канд. техн. наук, НТУ "ХПИ",
П.А. ШИТЬКОВ, НТУ "ХПИ"

ИСПОЛЬЗОВАНИЕ ЯЗЫКА *XML* ДЛЯ ПРОМЕЖУТОЧНОГО ПРЕДСТАВЛЕНИЯ ПРОГРАММЫ

Рассмотрены основные виды унифицированного промежуточного представления данных для виртуальных машин, что дает возможность создания кроссплатформенных программных продуктов. Предложен и обоснован способ реализации формата промежуточного представления программной модели на основе языка *XML*.

Ключевые слова: промежуточное представление данных, кроссплатформенные программные продукты, виртуальная машина, язык *XML*.

Постановка проблемы. Работоспособность любой програмы ограничена рамками операционной системы, для которой она создается. Классические архитектуры компиляторов не предполагают переноса платформ, мало того, порой работоспособность программы нарушается при смене версии какого-либо её компонента. Основным толчком к созданию технологии, позволяющей не привязываться к конкретной платформе, стало появление больших вычислительных машин, вмещающих в себе несколько операционных систем. Здесь возникали очевидные неудобства при подборе программного обеспечения. Программы приходилось перекомпилировать под различные платформы, а значит затрачивать значительное количество интеллектуальных и материальных ресурсов. Поэтому на сегодняшний день доминирующей идеей при построении компиляторов является отказ от привязки к конкретной платформе [1].

Использование виртуальных машин в области исполнения программных моделей открывает возможность создания кроссплатформенных программных продуктов. Это достигается за счет компиляции исходного текста программы в унифицированное промежуточное представление, которое и является входными данными для виртуальной машины [2, 3].

Анализ литературы. Как показывает анализ существующих реализаций промежуточного представление программ, наибольшее распространение получило использование двоичного представления программ [4]. Самая сильная сторона двоичного представления – отсутствие избыточности полученного кода. Также бинарный формат обеспечивет защиту кода, так как он зачастую не документирован, а без знания формата файла, данные из него получить раньше было практически невозможно. Однако появление программ декомпиляторов, которые производят обратное преобразование кода программы в исходный код, лишили бинарный формат этого преимущества. Необходимо отметить и значительные трудовые затраты на разработку

двоичного формата файла.

Байт-код или байткод (англ. *byte-code*), иногда также используется термин псевдокод – машинно-независимый код низкого уровня, генерируемый транслятором и исполняемый интерпретатором. Большинство инструкций байт-кода эквивалентны одной или нескольким командам ассемблера. Трансляция в байт-код занимает промежуточное положение между компиляцией в машинный код и интерпретацией. Программа на байт-коде обычно выполняется интерпретатором байт-кода. Преимущество байт-кода заключается в его портируемости, т. е. один и тот же байт-код может исполняться на разных платформах и архитектурах. То же самое преимущество дают интерпретируемые языки. Однако, поскольку байт-код обычно менее абстрактный, более компактный и более "компьютерный" чем исходный код, эффективность байт-кода обычно выше, чем чистая интерпретация исходного кода, предназначенного для правки человеком. По этой причине многие современные интерпретируемые языки на самом деле транслируют в байт-код и запускают интерпретатор байт-кода. К таким языкам относятся *Perl*, *PHP* и *Python*. Программы на *Java* [5, 6] обычно передаются на целевую машину в виде байт-кода, который перед исполнением транслируется в машинный код "на лету" – с помощью *JIT*-компиляции. В стандарте открытых загрузчиков *Open Firmware* фирмы *Sun Microsystems* байт код представляют операторы языка *Forth*.

Промежуточный код, сгенерированный компилятором, представляется также языком *MSIL* (*Microsoft Intermediate Language*), который разработан фирмой *Microsoft* для платформы *.NET Framework* [7]. Код на *MSIL* генерируют, в частности, все компиляторы для платформы *.NET* самой фирмы Майкрософт, входящие в среду разработки *Visual Studio* (*C#, Managed C++*, *Visual Basic .NET, Visual J# .NET*) [8, 9]. Язык *MSIL* по синтаксису и мнемонике напоминает ассемблер. Его можно рассматривать как ассемблер виртуальной машины *.NET*. В то же время язык *MSIL* содержит некоторые достаточно высокоуровневые конструкции, повышающие его уровень по сравнению с ассемблером для любой реально существующей машины, и писать код непосредственно на *MSIL* легче, чем на ассемблере для реальных машин. Поэтому его можно рассматривать как своеобразный "высокоуровневый ассемблер". *MSIL* сохраняет достаточно много информации об именах, использованных в исходной программе: имена классов, методов и исключительных ситуаций сохраняются и могут быть извлечены при обратном ассемблировании. Однако извлечь из *MSIL* сопровождаемые исходные тексты вряд ли возможно, так как имена локальных переменных, констант и параметров сохраняются отдельно, в файле *.PDB* и доступны только разработчику для отладки.

Цель статьи. Разработка собственного формата промежуточного представления программной модели.

В качестве собственного формата промежуточного представления программы предложено применение очень распространенного формата для представления данных – языка *XML*. За долгое время существования языка *XML* он получил всеобщее признание и применяется практически в любой программе, требующей сохранения каких-либо данных.

XML (англ. *eXtensible Markup Language*) – расширяемый язык разметки, рекомендованный Консорциумом Всемирной паутины, является текстовым форматом, предназначенным для хранения структурированных данных (взамен существующих файлов баз данных), для обмена информацией между программами, а также для создания на его основе более специализированных языков разметки (например, *XHTML*), иногда называемых словарями. *XML* является упрощённым подмножеством языка *SGML*. Язык *XML* обеспечивает совместимость при передаче структурированных данных между разными системами обработки информации, особенно при передаче таких данных через Интернет. Словари, основанные на *XML* (например, *RDF, RSS, MathML, XHTML, SVG*), сами по себе формально описаны, что позволяет программно изменять и проверять документы на основе этих словарей, не зная их семантики, то есть не зная смыслового значения элементов. Важной особенностью *XML* также является применение так называемых пространств имён (англ. *namespace*).

Проанализируем сильные и слабые стороны данного решения.

Достоинства.

1. *XML* – человеко-ориентированный формат, одновременно понятный и человеку, и компьютеру. Как говорилось выше, защищать промежуточный код путем усложнения представления данных не является тривиальной и дешевой задачей, которая еще и не решает основной проблемы безопасности.
2. *XML* поддерживает *Unicode*. Это даст нам возможность без труда интегрировать поддержку разноязычных имен переменных в программе.
3. В формате *XML* могут быть описаны основные структуры данных – такие как записи, списки и деревья. Программу можно представлять в виде дерева.
4. *XML* имеет строго определённый синтаксис и требования к анализу, что позволяет ему оставаться простым, эффективным и непротиворечивым.
5. *XML* – формат, основанный на международных стандартах. Это дает предпосылку, что наше предложение может стать новым стандартом решения данной проблемы.
6. *XML* не зависит от платформы, так как является стандартом.
7. *XML* является подмножеством *SGML* (который используется с 1986 года). Уже накоплен большой опыт работы с языком и созданы специализированные приложения. Это подразумевает наличие большой инструментальной базы для работы с *XML*.

Недостатки.

1. Синтаксис *XML* избыточен.

2. Размер документа *XML* примерно на порядок больше бинарного представления тех же данных.

3. Размер документа *XML* существенно больше, чем документа в альтернативных текстовых форматах передачи данных (например, *JSON, YAML*) и особенно в форматах данных, оптимизированных для конкретного случая использования.

4. Избыточность *XML* может повлиять на эффективность приложения.

5. Возрастает стоимость хранения, обработки и передачи данных.

6. *XML* не содержит встроенной в язык поддержки типов данных. В нём нет понятий "целых чисел", "строк", "дат", "булевых значений" и т. д.

С помощью языка *XML* любой текс программы может быть представлен в промежуточной форме в виде синтаксического дерева. Такое преобразование программы является задачей с небольшим уровнем сложности. Это дает право считать использование *XML* применимым к широкому кругу задач построения входных моделей, так как в формате *XML* могут быть описаны основные конструкции языков программирования и структуры данных.

Рассмотрим примеры представления программных структур с использованием языка *XML,* приведенные в таблице. Примеры представлены для языка C#.

Как видно из примеров трансформации стандартных программных конструкций в *XML*, применение *XML* достаточно для решения данного класса задач. За счет избыточности формата, наблюдается увеличение объема данных. Этот недостаток является существенным и может быть устранен путем последующего применения алгоритмов сжатия. Так как формат является текстовым, то применение алгоритмов сжатия очень эффективно. При использовании алгоритма сжатия *DEFLATE* (алгоритм сжатия без потерь, который использует комбинацию алгоритма *LZ*77 и алгоритма Хаффмана) получаем следующие результаты:

Начальный размер файла: 60000 байт.

Размер сжатого файла: 1340 байт.

При этом размер файла уменьшился в 45 раз.

Выводы. В данной статье был предложен и обоснован способ реализации формата промежуточного представления программной модели на основе языка *XML*. Конструктивные особенности языка *XML* являются достаточными для представления древовидных программных образований любой сложности. В формате *XML* могут быть описаны основные конструкциии языков программирования и структуры данных. Это позволяет считать, что его использование может стать достойной альтернативой при решении подобных задач. Так как данный формат является избыточным, то для устранения этого недостатка предложено применять алгоритмы сжатия, что, по результатам тестирования, приводит к существенному уменьшению размера файла.

Таблица

Примеры представления программных структур с использованием языка XML

C#	XML
Процедура/Функция	

`int MyProc(int a, string s) {`
`}`
	`<Procedure name = "MyProc">`
	`  <Params>`
	`  </Param name = "a", type = "int">`
	`  </Param name = "s", type = "string">`
	`  </Params>`
	`  <Vars>`
	`  </Var name = "a" type = "int" val = "0">`
	`  </Var name = "s" type = "string" val = "0">`
	`  </Var name = "rez" type = "int" val = "0">`
	`  </Vars>`
	`  <Pop val = "s">`
	`  <Pop val = "a">`
	`  // Program code`
	`  <Push val = "rez">`
	`</Procedure>`

Вызов Процедуры	

`int MyProc(int a, string s) {`
`  return Calc(a);`
`}`
	`<Procedure name = "MyProc">`
	`  <Params>`
	`  </Param name = "a", type = "int">`
	`  </Param name = "s", type = "string">`
	`  </Params>`
	`  <Vars>`
	`  </Var name = "a" type = "int" val = "0">`
	`  </Var name = "s" type = "string" val = "0">`
	`  </Var name = "rez" type = "int" val = "0">`
	`  </Vars>`
	`  <Pop val = "s">`
	`  <Pop val = "a">`
	`  <Call method = "Calc" param1 = "a">`
	`  <Push val = "rez">`
	`</Procedure>`

Математическое выражение	

`int a = 8;`
`int x = 5 + a * 3;`
	`<Vars>`
	`  </Var name = "x" type = "int" val = "0">`
	`  </Var name = "a" type = "int" val = "8">`
	`  </Var name = "local1" type = "int" val = "5">`
	`  </Var name = "local2" type = "int" val = "3">`
	`</Vars>`
	`  <Mul param1 = "a" param2 = "local2">`
	`  <Pop val = "x">`
	`  <Add param1 = "x" param2 = "local1">`
	`  <Pop val = "x">`

Конструкция if	

`if(x==0)`
`  x=1;`
`else`
`  x=2;`
	`<Vars>`
	`  </Var name = "x" type = "int" val = "0">`
	`  </Var name = "local1" type = "int" val = "1">`
	`  </Var name = "local2" type = "int" val = "2">`
	
	`</Vars>`
	`<Check param1 = "x", param2 = "0", mode = "equals" elseLab=`
	`"ElseLab1">`
	`  </Mov val1 = "x", val2 = "local1";`
	`</Check>`
	`<Label name = "ElseLab1">`
	`  Mov val1 = "x", val2 = "local2";`
	`</Label>`

Список литературы: 1. *Хантер Р.* Основные концепции компиляторов. – М.: Вильямс, 2002. – 256 с. **2.** *Harel D.* Biting the silver bullet: Toward a brighter future for system development // Computer. – 1992. – Jan. – С. 8 – 20. **3.** *Гавриленко С.Ю., Шитьков П.С.* К вопросу о генерации кода в современных компиляторах // Вестник НТУ "ХПИ". Тем. вып. "Автоматика и приборостроение". – Харьков: НТУ "ХПИ". – 2007. – № 10. – С. 20 – 26. **4.** *Ульман Джеффри, Ахо Альфред, Сети Рави.* Компиляторы: принципы, технологии и инструменты. – М: Вильямс, 2001. – 768 с. **5**. *Савитч У.* Язык Java. Курс программирования. – М.: Вильямс, 2002.– 928 с. **6.** *Шилдт Г., Холмс Д.* Искусство программирования на Java. – М.: Вильямс, 2005. – 336 с. **7**. *Байдачній С.С.* NET Framework 2. 0. Секреты создания Windows-приложений (Подготовка к сдаче экзамена для получения статуса MCSD.NET). – М.: Солон, 2006. – 520 с. **8**. *Нейгел К.* Язык программирования C# 2005 для профессионалов. – М.: Вильямс, 2002. – 1376 с . **9**. *Лавров С.С.* Основные понятия и конструкции языков программирования. – М.: Финансы и статистика, 1982.– 227 с.

УДК 681.3.06

Використання мови XML для проміжного подання програми / Гавриленко С.Ю., Шитьков П.О. // Вісник НТУ "ХПІ". Тематичний випуск "Інформатика і моделювання". – Харків: НТУ "ХПІ", 2008. – № 24. – С. 19 – 24.

Розглянуті основні види уніфікованого проміжного подання даних для віртуальних машин, що дає змогу створення кросплатформених програмних продуктів. Запропоновано та обосновано спосіб реалізації формата проміжного подання програмної моделі на основі мови XML. Табл.: 1. Бібліогр.: 9 назв.

Ключові слова: проміжне подання даних, кросплатформені прорамні продукти, віртуальна машина, мова XML.

UDK 681.3.06

Use of language XML for intermediate representation of the program / Gavrilenko S.J., Shitkov P.S. // Herald of the National State University "KhPI". Subject issue: Information science and modelling. – Kharkov: NSU "KhPI", 2008. – № 24. – P. 19 – 24.

The basic kinds of the unified intermediate data presentation for virtual machines, that enables creations crossplatform software products are considered. The way of realization of a format of intermediate representation of program model on the basis of language XML is offered and proved. Tabl.: 1. Refs: 9 titles.

Key words: intermediate data presentation, crossplatform software products, virtual machine, language XML.

Поступила в редакцию 07.05.2008