МОДЕЛИ ОБРАБОТКИ ИНФОРМАЦИИ И ПРОГРАММНЫЕ СРЕДСТВА ДЛЯ УНИВЕРСАЛЬНЫХ МОДЕЛЕЙ РЕШЕНИЯ ЗАДАЧ

Ивашенко В. П.
Кафедра интеллектуальных информационных технологий, Факультет информационных технологий и управления, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: ivashenko@bsuir.by

В статье рассматриваются универсальные модели обработки информации, операции и команды, а также построенная на их основе программная модель машины Тьюринга

Введение

В системах, ориентированных на решение задач [1], используются разные модели и формализмы [2], направленные на решение задач заданного класса, среди которых можно выделять обеспечивающие решение задач менее или более широких классов [3]. Особый интерес представляют универсальные модели [3], способные обеспечить интеграцию различных моделей решения задач [2]. Многие из этих моделей представляют в основном теоретический интерес [4], так как не всегда удобны или пригодны для практического использования, однако, нет оснований отрицать, что практическое использование подобных моделей в интеллектуальных системах решения задач является востребованным и целесообразным. Вместе с тем, следует отметить, что зачастую при практической реализации формализмов и моделей, которые в соответствии со спецификацией полагаются универсальными, упускаются из виду важные технические аспекты, вследствие чего универсальность этих моделей остаётся существовать только теоретически, а практически пользователь получает ограниченную и неуниверсальную систему, при достижении пределов возможностей которой приходится начинать разработку новой системы или её версии, требующую существенных трудозатрат, связанных в том числе с переносом данных и задач пользователя.

I. Модели обработки информации, ориентированные на работу с памятью

Предложенные в [5] логические и арифметические модели обработки информации (МОИ), являются конечными и вследствие этого, несмотря на полноту, неуниверсальными. Для того, чтобы обеспечить универсальность, эти модели расширяются до модели, в которой поддерживаются операции адресной арифметики и адресного доступа к линейно адресуемой памяти, которая полагается неограниченной именно с целью обеспечения универсальности. На практике существуют объективные технические ограничения, которые не позволяют реализовать бесконечную память или легко обеспечить её неограниченность, достраивая её «на лету». Однако, известны подходы, решения, позволяющие существенно минимизировать перепрограммирование и затраты на перенос данных и задач в системах, допускающих масштабирование [6], которое в рассматриваемом случае касается информационного объёма памяти. Эти подходы и решения связаны с организацией системы команд, которые абстрагируются от особенностей текущей физической реализации вычислительной системы. Таким образом, команды системы команд адресной арифметики и адресного доступа к памяти должны быть абстрагированы от реализационных ограничений на информационный объём физически доступной в текущий момент памяти.

Система команд адресного доступа, строится на основе модели адресного доступа к памяти, которая состоит из n ($\leq 2^d$) адресных регистров неограниченной разрядности, и линейно упорядоченного набора ячеек памяти, которая также двусторонне неограничена. Однако, при конкретной реализации допускается использовать ограниченное количество разрядов (не меньшее чем значение d) и ячеек памяти (не менее чем 2^d): младшие d разрядов любого адресного регистра являются аргументами логических и арифметических операций, которые вместе с операциями адресных арифметики и доступа входят в объединённую систему команд. В случае, когда разрядов адресных регистров или ячеек памяти становится недостаточно, процесс вычислений прерывается до тех пор, пока не будет подготовлена реализация с достаточным количеством требуемых ресурсов. При прерывании состояние машины сохраняется, это сохранённое состояние впоследствии загружается (копируется) на подготовленную реализацию и вычислительный процесс возобновляется, при этом не требуется вносить никаких изменений ни в программы, ни в данные. Для того, чтобы отследить когда необходимо прервать процесс, в каждую реализацию вводится пара значений *minor* и *major*, с учётом которых строится реализация операций адресной арифметики: увеличение и уменьшение значений адресных регистров (см. табл. 1 и 2), максимум, минимум, операции условного перехо-

да и другие адресные операции. Модель вычислений предполагает, что команды хранятся наравне с данными в общей памяти, поэтому один из адресных регистров (нулевой) используется в качестве адресного регистра исполняемой команды.

Таблица 1 – Команда вида *add_pointer*

$\langle\#1\rangle\backslash P[\#2]$	$\leq major-\langle\#1\rangle$	$>major-\langle\#1\rangle$
$\leq major-P[\#2]$	$P[\#2]+=\langle\#1\rangle$	-
$>major-P[\#2]$	-	прерывание; сохранение состояния

Таблица 2 – Команда вида *sub_pointer*

$\langle\#1\rangle\backslash P[\#2]$	$<minor+\langle\#1\rangle$	$\geq minor+\langle\#1\rangle$
$\leq P[\#2]-minor$	-	$P[\#2]-=\langle\#1\rangle$
$>P[\#2]-minor$	прерывание; сохранение состояния	-

II. Программирование универсальной машины Тьюринга

Чтобы показать универсальность совокупности предложенных моделей обработки информации, рассмотрим реализацию на их основе такой вычислительной системы, как машина Тьюринга (МТ). Для чего требуется ответить на несколько вопросов: «как запрограммировать (универсальную) машину Тьюринга?», «как распределить память и обеспечить доступ к бесконечной ленте?», «как организовать хранение программы МТ в памяти?». Программирование осуществляется с помощью предложенной системы команд. Память распределяется блочным образом (см. рис. 1—3). Хранение команд программы МТ (см. рис. 4) сводится к хранению проверяемого и записываемого символов (0 или $2^{32}-1$), направления перехода к следующей ячейке (0, 1 или $2^{32}-1$) и смещения (занимает необходимое количество ячеек) до блока команд (следующего внутреннего состояния) МТ от начального.

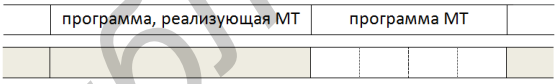

Рис. 1 – Общий вид, разбиение памяти на блоки (вверху) и блочное хранение программы МТ (внизу)

Рис. 2 – Блочное хранение команд программы МТ

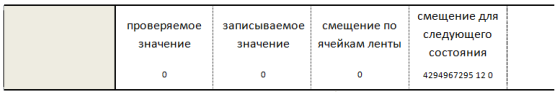

Рис. 3 – Структура блока команды программы МТ

```
0 0     0     2³²−1 132  0  2³²−1 2³²−1   1       2³²−1 12  0
0 0     1     2³²−1 24   0  2³²−1 2³²−1   1       2³²−1 96  0
0 2³²−1 2³²−1 2³²−1 36   0  2³²−1 2³²−1   1       2³²−1 24  0
0 0     2³²−1 2³²−1 48   0  2³²−1 2³²−1   2³²−1   2³²−1 36  0
0 0     2³²−1 2³²−1 48   0  2³²−1 2³²−1   2³²−1   2³²−1 60  0
0 0     1     2³²−1 12   0  2³²−1 2³²−1   1       2³²−1 72  0
0 0     1     2³²−1 84   0  2³²−1 0       1       2³²−1 72  0
0 0     1     2³²−1 84   0  2³²−1 2³²−1   1       2³²−1 24  0
0 2³²−1 1     2³²−1 96   0  2³²−1 2³²−1   2³²−1   2³²−1 108 0
0 0     0     2³²−1 132  0  2³²−1 0       1       2³²−1 120 0
0 2³²−1 0     2³²−1 132  0  2³²−1 2³²−1   1       2³²−1 120 0
2³²−1 2³²−1
```

Рис. 4 – Код программы МТ добавления единицы к строке единиц и её данные

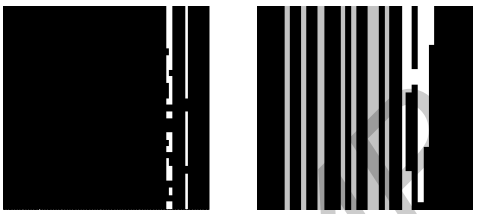

Рис. 5 – История состояния регистров (слева) и фрагмент истории состояний памяти (справа) работающей МТ (данные изображены белым)

Заключение

Важным является то, что объединённая система команд, способная обеспечить реализацию универсальных вычислительных систем, имеет в своём наборе команды, которые соответствуют востребованным и практически используемым командам в современных компьютерных архитектурах, что позволяет легко совмещать классическое программирование и поддержку универсальных моделей решения задач.

Система команд и демонстрационный пример МТ (см. рис. 5 и рис. 6) реализованы средствами JavaScript (https://bitbucket.org/version/openjsvvm/).

1. Wolfram, S. Jeopardy, IBM, and Wolfram|Alpha. Mode of access: http://blog.stephenwolfram.com/2011/01/jeopardy-ibm-and-wolframalpha/ Date of access: 17.09.2017.
2. Luger, G. Stubblefield, William (2004), Artificial Intelligence: Structures and Strategies for Complex Problem Solving (5th ed.), The Benjamin/Cummings Publishing Company, Inc., p. 720.
3. Wolfram, S. A New Kind of Science. Champaign, IL: Wolfram Media, Inc. 2002. p. 1197.
4. Zaitsev D. A. Toward the Minimal Universal Petri Net, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2014, Vol. 44, No. 1, pp. 47—58.
5. Ивашенко, В. П. Модели обработки информации в интеллектуальных системах, основанных на семантических технологиях / В. П. Ивашенко, А. С. Бельчиков, А. П. Еремеев // Информационные технологии и системы 2016 (ИТС 2016) : материалы международной научной конференции (БГУИР, Минск, Беларусь, 26 октября 2016) / редкол. : Л. Ю. Шилин [и др.]. — Минск: БГУИР, 2016. — С. 106—107.
6. FitzRoy-Dale, N. The VLIW and EPIC processor architectures, Master Thesis, New South Wales University, 2005.

Рис. 6 – История состояний памяти с работающей МТ