ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
им. М.В. КЕЛДЫША
РОССИЙСКОЙ АКАДЕМИИ НАУК

А.М.Горелик

СОВРЕМЕННЫЙ ФОРТРАН ДЛЯ КОМПЬЮТЕРОВ ТРАДИЦИОННОЙ АРХИТЕКТУРЫ И ДЛЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

(аналитический обзор)

Москва
2003

Работа подготовлена в соответствии с методическими рекомендациями Российского фонда фундаментальных исследований и поддержана этим фондом (проект 02-01-07012).

А.М.Горелик

Современный Фортран для компьютеров традиционной архитектуры и для параллельных вычислительных систем (аналитический обзор)

АННОТАЦИЯ

Рассматриваются проблемы стандартизации языка Фортран, дается обзор новых возможностей современных стандартов Фортрана. Приводится краткая информация о проекте будущего стандарта (Фортран 2000). Наибольшее внимание уделяется тем новым средствам языка, которые позволяют использовать современные технологии программирования, а также средствам поддержки параллельности, которые имеются непосредственно в действующем стандарте. Обсуждаются языки для параллельных ЭВМ, являющиеся расширениями Фортрана.

A.M.Gorelik

Modern Fortran for traditional and parallel architectures

The preprint of the Keldysh Institute of Applied Mathematics,
Russian Academy of Sciences

ABSTRACT

An overview of new features of modern FORTRAN standards and an outline of the future FORTRAN standard (FORTRAN 2000) are given. FORTRAN-based programming languages for parallel computers are discussed.

This paper was prepared in accordance with methodical recommendations of Russian Foundation for Fundamental Investigations and was supported by this Foundation (Project 02-01-07012).

СОДЕРЖАНИЕ

А. Анализ проблематики исследований

Предисловие

Язык Фортран занимает лидирующее положение среди языков программирования, ориентированных на решение научно-технических задач, требующих большого объема вычислений. Особенно актуальным является применение Фортрана при решении крупномасштабных вычислительных задач с использованием современных суперЭВМ. Решение таких задач требуется в различных сферах фундаментальных научных исследований и во многих прикладных областях.

Фортран постоянно развивается и совершенствуется в соответствии с развитием вычислительной техники, языков и технологии программирования. Одной из наиболее важных причин популярности и живучести Фортрана является огромный фонд прикладных программ, который накоплен за десятилетия существования языка. Этому в немалой степени способствовали удобство и эффективность выполнения программ, а также стандартизация языка на международном уровне.

Разработка новых мобильных средств языка является результатом фундаментальных исследований (выполняемых специалистами многих стран) в области теории и практики программирования, в области развития архитектуры современных вычислительных систем и технологии программирования, а также является результатом анализа и обобщения опыта использования вычислительной техники при решении задач, требующих большого объема вычислений.

Бытующее мнение о том, что Фортран устарел – ошибочно. Конечно, в связи с широким внедрением персональных ЭВМ во многие сферы человеческой деятельности, удельный вес Фортрана в общем объеме программного обеспечения снизился. Однако при решении больших вычислительных задач предпочтение отдается современному Фортрану.

Некоторый спад интереса к Фортрану был вызван большой задержкой в разработке Фортрана 90. Однако в настоящее время после завершения разработки, вступления в действие современных международных стандартов Фортрана (Фортран 90 и Фортран 95) и реализации их практически для всех вычислительных систем, а также в связи с использованием высокопроизводительных параллельных компьютеров, интерес к Фортрану вновь возрос.

Предлагаемый обзор посвящен анализу современного состояния и перспективам развития языка Фортран.

В первой части обзора рассматриваются проблемы стандартизации языка, анализируются новые (по сравнению с Фортраном 77) возможности действующего международного стандарта (Фортран 95) и обсуждаются перспективы дальнейшего развития языка. Наибольшее внимание уделяется тем новым средствам языка, которые позволяют использовать современные технологии программирования, а также средствам поддержки параллельности, которые имеются непосредственно в действующем стандарте.

Вторая часть обзора посвящена языкам для параллельного программирования, основанным на стандарте Фортрана.

Часть I
Современный международный стандарт языка Фортран

1. Введение

Действующий международный стандарт Фортрана был принят в 1997 году; неформальное название его – Фортран 95 [1]. Язык является относительно небольшим расширением предыдущего стандарта - Фортран 90 [2, 3], который был утвержден в 1991 году и пока не потерял своей актуальности.

Фортран 90/95 существенно расширяет возможности своих предшественников. В то же время практически сохраняется преемственность с предыдущими стандартами языка, что позволяет использовать ранее созданный фонд прикладного программного обеспечения. Фортран 90/95 позволяет создавать более мобильные и более надежные программы по сравнению с Фортраном 77 и обеспечивает современный стиль программирования.

В современные стандарты включены средства, позволяющие расширить сферу применения языка, использовать современные технологии программирования, эффективно реализовать вычислительные алгоритмы. При разработке языка учтены новые архитектурные решения современных вычислительных систем.

Следует отметить, что для небольших программ не очень существенно, какая методология программирования выбрана, и какой язык используется. При разработке больших программ применение новых технологий (что обеспечивает современный Фортран) дает существенный выигрыш. В то же время программисты, использующие Фортран 77 (который был разработан еще в 70-е годы прошлого века) и даже более ранние версии, вынуждены использовать старые технологии. Переход

к новым технологиям с сохранением многолетнего вклада в разработку Фортран-программ – это требование времени. Поскольку Фортран 90/95 является расширением Фортрана 77, некоторые из новых концепций могут быть реализованы путем модификации старых программ.

Помимо официального описания стандарта языка [1, 2], выпущены десятки книг на разных языках (по моим сведениям на 11), которые содержат неформальное описание языков Фортран 90 и Фортран 95. Во многих университетах мира читаются лекции, организованы разнообразные курсы по изучению современного Фортрана. На русском языке книг, содержащих описание стандарта языка Фортран 95, пока нет. Для Фортрана 90 выпущен перевод официального описания стандарта [4] и книги [5, 6] с неформальным описанием этого стандарта. О реализациях современных стандартов будем говорить позже (см. 12).

Исследования, связанные с новыми возможностями современных стандартов Фортрана, проводятся как за рубежом, так и в нашей стране (проекты РФФИ № 97-01-0361 и 00-01-00043 [7-13]).

2. Стандартизация языка

2.1. Цели стандартизации языка

Стандартизация языков программирования создает предпосылки для повышения мобильности программного обеспечения для компьютеров любой архитектуры. Стандартизация Фортрана является одной из причин долгожительства языка, так как именно благодаря стандартизации обеспечена возможность использования огромного фонда прикладных программ, которые созданы за десятилетия существования языка.

Язык подвергался стандартизации в рамках ANSI и ISO четыре раза (Фортран 66, Фортран 77, Фортран 90, Фортран 95). В настоящее время на международном уровне ведется работа по дальнейшему развитию языка: разрабатывается проект языка Фортран 2000 (завершение планируется в 2004г.), который предусматривает весьма существенные нововведения.

Вопросам стандартизации языков программирования, в т.ч. и Фортрана, во многих странах уделяется большое внимание, в этой деятельности участвуют ведущие фирмы, научные и учебные центры. К сожалению, в нашей стране роль стандартов на языки программирования явно недооценивается, и новое программное обеспечение часто создается без учета международных стандартов. Это касается как разработчиков прикладных программ, так и разработчиков

системного программного обеспечения. Опыт показал, что игнорирование международных стандартов приводит к большим затратам при адаптации программ к другой вычислительной среде.

Следует отметить, что стандарты Фортрана разрешают включать в реализацию языка дополнительные средства, не предусмотренные стандартом. При этом современные стандарты Фортрана требуют, чтобы компиляторы выдавали сообщения о несоответствии стандарту, об используемых расширениях. Программист, использующий дополнительные средства, должен понимать, что они могут отсутствовать при переносе его программы в другую вычислительную среду.

Многие разработчики программного обеспечения четко выделяют такие расширения в документации. Однако в литературе, содержащей описание конкретных реализаций, к сожалению, не всегда четко разграничивается, какие черты являются нестандартными, что затрудняет разработку мобильных программ. Программист, который воспользуется подобным описанием, может попасть в затруднительное положение. Например, типичная ситуация, когда отладив программу на персональном компьютере, расчеты производятся с помощью другой вычислительной техники (например, с помощью какой-либо параллельной вычислительной системы). Для того чтобы избежать подобных проблем программисту необходимо знание именно стандарта и предпочтение надо отдавать средствам, предусмотренным в стандарте.

Еще больше проблем для создания мобильных программ представляют реализации, которые не соответствуют стандарту. Если прикладная программа ориентирована на такую реализацию, то затраты на ее переделку для использования в другой вычислительной среде в некоторых случаях могут быть сопоставимы с разработкой новой программы.

2.2. Кто и как разрабатывает международные стандарты языка Фортран

Международные стандарты языка являются результатом совместной деятельности экспертов многих стран. Ответственным за стандартизацию языков программирования на международном уровне является подкомитет 22 (SC22), входящий в состав Объединенного технического комитета (JTC1) Международной организации по стандартизации (ISO) и Международной электротехнической комиссии (IEC).

Непосредственной работой по стандартизации языков программирования, поддержкой уже действующих стандартов и разработкой общих требований к

стандартам на языки программирования занимаются эксперты разных стран, объединенные в соответствующие рабочие группы подкомитета SC22.

Стандартизацией языка Фортран занимаются Американский технический комитет J3 ANSI и эксперты рабочей группы WG5 (указанного подкомитета). Членами WG5 являются специалисты многих стран, в т.ч. и нашей страны. В их числе представители компьютерных фирм, крупных университетов. Многие из тех, кто ответствен за разработку коммерческих Фортран-компиляторов, являются членами J3 и/или WG5.

Представители национальных рабочих групп и все заинтересованные специалисты имеют возможность присылать свои предложения, комментарии, замечания. По результатам международного обсуждения всех поступивших предложений и дальнейшего голосования принимаются все принципиальные решения.

Многие вопросы обсуждаются заочно, путем обмена информацией по электронной почте. По электронной почте также производится неформальное голосование. Примерно один раз в год рабочая группа WG5 собирается на совещание для обсуждения текущих вопросов и выработки соответствующих решений.

3. Обзор новых возможностей современного Фортрана

3.1. Структура стандарта

Современные стандарты Фортрана представляют собой семейство стандартов, состоящее из нескольких частей. Действующий стандарт Фортран 95 состоит из трех частей. Первая часть - основной (базовый) язык [1]. Остальные части являются дополнительными. При этом не требуется, чтобы компилятор, соответствующий стандарту, обязательно реализовывал дополнительные части. Вторая часть стандарта [14] содержит описание средств для работы с символьными строками переменной длины. Третья часть Фортрана 95 [15] определяет описание языка условной компиляции.

Ниже приводится краткий обзор новых по сравнению с Фортраном 77 средств базового языка. Там, где специально не оговорено, эти новшества имеются и в Фортране 90, и в Фортране 95.

3.2. Свободный формат исходной программы и имена

В Фортране 90/95, наряду с фиксированным форматом исходного текста программы, разрешен свободный формат. Свободный формат допускает помещение более одного оператора в строке, при этом в качестве разделителя используется точка с запятой. Признак продолжения оператора на строку продолжения - символ & - указывается в конце той строки, которую надо продолжить. Комментарии записываются после символа восклицательный знак в начале строки или в любой позиции строки после оператора. В свободном формате пробелы являются значащими.

Имена могут содержать до 31 символа; в именах допускается символ подчеркивания. Можно использовать строчные латинские буквы, которые всюду, кроме символьных констант и символьных спецификаций формата, считаются равнозначными соответствующим прописным буквам.

3.3. Параметризованные типы данных, атрибуты и средства контроля соответствия типов

В Фортране 90/95 свойства объектов могут описываться в соответствующих операторах спецификации атрибутов (как и прежде), либо с помощью атрибутов в операторе объявления типа. В одном операторе объявления типа можно описать несколько атрибутов объектов, перечисленных в данном операторе после двойного двоеточия; в этих операторах допускается также инициализация объектов.

Примеры

```
REAL, DIMENSION (3), PARAMETER :: VECTOR = ( / 1., 2., 3. /)
REAL, SAVE :: XY (10, 10)
```

Для всех встроенных типов введены (необязательные) средства параметризации, т.е. каждый встроенный тип может иметь несколько разновидностей. Параметризация встроенных типов обеспечивает возможность задания способа представления данных, что позволяет компиляторам поддерживать короткие целые, более двух видов точности для вещественных и комплексных данных, а также многобайтовые символьные данные (для языков с большим набором символов, таких как китайский или японский) или использовать дополнительные наборы символов для конкретных приложений.

Приведенный ниже пример иллюстрирует способ задания точности и диапазона для вещественных данных.

Пример

```
!  Два варианта для задания способа представления, обеспечивающего
!  точность не менее 10 знаков и диапазон десятичного порядка не менее
!  чем (-30, +30)
      INTEGER, PARAMETER :: NUMS = SELECTED_REAL_KIND (10, 30)
      REAL (KIND = NUMS) R1, R2          ! NUMS - параметр типа
      REAL (SELECTED_REAL_KIND (10, 30)) :: R1, R2
```

Для приведенного примера значения переменных R1 и R2 на одних компьютерах могут отображаться как обычные вещественные числа, занимающие одно машинное слово, на других - как "длинные" (двойной точности) вещественные числа. Однако при переносе данной программы на компьютер с иной формой представления данных не требуется менять ни описания переменных R1 и R2, ни другие операторы, связанные с этими переменными, так как в любой реализации (если компилятор не выдал отказа) должны быть обеспечены соответствующие точность и диапазон. Таким образом, указанные средства больше отвечают требованиям мобильности программ, чем использование непараметризованных традиционных для Фортрана типов REAL и DOUBLE PRECISION.

Для языка Фортран традиционно был характерен принцип умолчания, т.е. автоматическое приписывание объектам типа, если тип объекта не указан явно. В Фортран 90/95 введен оператор IMPLICIT NONE. При наличии такого оператора типы объектов в данной программной единице должны быть объявлены явно. Явное объявление объектов обеспечивает возможность контроля типов, что, очевидно, полезно для повышения надежности создаваемых программ. IMPLICIT NONE является более гибким инструментом, чем просто обязательность описаний, что имеет место в других языках.

3.4. Типы и операции, определяемые пользователем

Введены средства, позволяющие определить новые типы данных - производные типы (структуры данных), которые представляют собой совокупность компонент. Компоненты производного типа могут иметь встроенный или ранее описанный производный тип; компонентами могут быть скаляры, массивы и указатели. Компонент производного типа может иметь такой же тип как описываемый, если он является указателем (т.е. имеет атрибут POINTER). Это полезно для создания связанных списков.

Скалярный объект производного типа представляет собой структуру. Допускаются массивы объектов производного типа. Компоненты объекта производного типа могут использоваться в обычных выражениях и операторах. Для объектов одинакового производного типа определена операция присваивания; кроме того, такие объекты могут использоваться в операторах ввода/вывода, в качестве параметров процедур и как результат функции.

Примеры

```
! Описание типа PERSON
TYPE PERSON
  INTEGER AGE
  CHARACTER (LEN = 10) NAME
END TYPE PERSON

! Объявление объектов описанного типа
TYPE (PERSON):: STUDENT, TEACHER, MEMBER (10)

! Операция над компонентами производного типа
N = TEACHER%AGE - STUDENT%AGE

! Компонент элемента массива производного типа
MEMBER (1) % AGE = 50

! Присваивание объекту производного типа
STUDENT = (20, '     SMITH')

! Пример связанного списка
TYPE LINK
  REAL VALUE
  TYPE (LINK), POINTER :: PREVIOUS
  TYPE (LINK), POINTER :: NEXT
END TYPE LINK
```

В языке имеется аппарат, позволяющий программисту распространить для производных типов встроенные операции (перегрузка) или описать новые операции для данных встроенных и производных типов. Эти возможности позволяют программисту определить абстрактные типы данных, что является одним из элементов объектно-ориентированного программирования.

3.5. Операции над массивами, секциями массивов и соответствующие присваивания. Операторы и конструкции WHERE и FORALL

В язык введен большой набор средств для работы с массивами и секциями массивов как с целыми объектами. Существенным является тот факт, что операции над массивами неявно специфицируют параллелизм действий над компонентами массивов (массива). Новые средства позволяют программисту описать алгоритмы обработки массивов в более лаконичной и наглядной форме (с элементами непроцедурности) чем при традиционном способе с использованием вложенных циклов и условных операторов. Кроме того, эти средства позволяют компилятору сгенерировать эффективный код с учетом особенностей аппаратуры, поскольку явно специфицируют векторные операции (см. 3.11.). Ниже рассматриваются некоторые из этих средств.

Секция массива обеспечивает возможность адресоваться к части массива. Секция задает последовательность индексов и тем самым выделяет часть массива, состоящую из элементов с соответствующими индексами. Секция массива может состоять не только из расположенных подряд элементов исходного массива, но также и из несвязанных областей этого массива.

Семантика операций и встроенных функций Фортрана 77 расширена таким образом, что их можно применять не только к скалярам, но и к массивам; при этом операции и стандартные функции выполняются поэлементно над соответствующими элементами массива. Операнды, участвующие в одной операции, должны быть согласованы по конфигурации (т.е. должны иметь одинаковую размерность и размер по каждому измерению; скаляр считается согласованным с любым массивом). Массив может быть также значением выражения и значением функций (как встроенных, так и определяемых пользователем).

Аналогичным образом расширена семантика операторов присваивания; результат операции такой, как если бы сначала выполнялись все операции в правой части, а затем - присваивание, которое может выполняться в произвольном порядке.

Примеры

```
REAL X (10, 20), Y (40), Z (-9:10), W (10)
!  Присваивание значений элементам 10-й строки матрицы X
X (10, :) = 2.8 * Y (2:40:2) + SQRT (Z)
!  Изменение порядка элементов массива на обратный
W (1:10) = W (10:1:-1)
```

Используя оператор WHERE или конструкцию WHERE-END WHERE, можно выполнить присваивание значений только некоторых элементов одного массива соответствующим элементам другого массива, т.е. выполнить присваивание под управлением логической маски.

Примеры

```
REAL  X(10, 20),  Z (-9:10),  V (20)
!  Вычисление логарифма для положительных элементов массива V
WHERE (V > 0.0 ) Z = LOG (V)
!  Конструкция WHERE
WHERE (V /= 0 )
  X (10, :) = 3.3 * Z / V
ELSEWHERE
  X (10, :) = 0.0
END WHERE
```

Одним из наиболее важных новшеств (в Фортране 95) является оператор и конструкция FORALL. FORALL допускает возможность специфицировать больший класс секций массивов, чем в Фортране 90. Например, диагональ массива в Фортране 95 может быть представлена с помощью FORALL.

Пример

```
FORALL (I = 1:N) A (I, I) = B(I)
```

FORALL функционально похож на цикл, но выполняется иначе. Так, оператор

```
FORALL (I = 2:N) C(I, I) = C(I-1, I-1)
```

дает результат, отличный от результата приведенной ниже последовательности операторов:

```
DO I = 2, N
  C(I, I) = C(I-1, I-1)
END DO
```

Это объясняется тем, что итерации DO-цикла выполняются последовательно так, что каждый следующий элемент диагонали модифицируется до его использования в следующей итерации. В противоположность этому, в FORALL все диагональные элементы выбираются и используются до сохранения

модифицированного значения в памяти. Заметим, что такой результат не зависит от того, поддерживает ли компилятор параллельное выполнение FORALL.

Имеется также конструкция FORALL – END FORALL. Заголовок конструкции может содержать логическую маску. Конструкции могут быть вложенными и могут содержать конструкции WHERE.

В языке имеется большой набор стандартных функций для работы с массивами. Функции редукции (ALL, ANY, COUNT, MINVAL, MAXVAL, PRODUCT, SUM) выполняют арифметические и логические операции над массивами. Имеются функции умножения матриц, транспонирования матрицы, вычисления скалярного произведения векторов, функции, выполняющие сдвиги позиций элементов массива, функции конструирования массива из элементов других массивов, различные справочные функции и др.

Пример

```
! Вычисление суммы положительных элементов массива X
C = SUM (X, MASK = X > 0.0)
```

3.6. Механизмы динамического размещения массивов

В современных стандартах введены следующие механизмы динамического размещения массивов:

- автоматические массивы, границы которых вычисляются при входе в процедуру;
- размещаемые массивы, границы которых вычисляются при выполнении программы;
- массивы, создаваемые в процессе выполнения программы, т.е. массивы, которые не были заранее явно объявлены в программе (см. 3.7.3.).

Эти средства позволяют пользователю организовать работу с массивами, размер которых заранее не известен, а также оперативно отводить память под массивы, требующиеся на том или ином этапе выполнения программы и освобождать память по мере необходимости.

Размещаемые массивы должны быть объявлены с атрибутом ALLOCATABLE; при объявлении размещаемого массива указывается только его размерность (ранг). При выполнении оператора размещения ALLOCATE вычисляются границы массива, его размер и отводится память. Память, занимаемая массивом, освобождается при выполнении оператора DEALLOCATE.

Примеры

```
SUBROUTINE  S (X, M)
COMMON  K
REAL, DIMENSION (M, 2:K) :: A                ! Автоматический массив A
REAL, DIMENSION, ALLOCATABLE :: Y(:)  ! Размещаемый одномерный массив Y
. . .
READ (*, *) N
ALLOCATE  Y (N)                              ! Размещение массива Y
```

3.7. Указатели

3.7.1. В язык введены указатели, с помощью которых можно ссылаться на объекты. С помощью указателей и аппарата производных типов можно создавать произвольные структуры данных: связанные списки, деревья и др. (см. 3.4). Использование указателей позволяет также создавать в процессе выполнения программы динамические объекты (см. 3.6 и 3.7.3).

Для объявления указателя используется атрибут POINTER. При объявлении объекта-указателя фактически специфицируются свойства (тип, размерность и т.п.) тех объектов, которые могут быть связаны с данным указателем; если указатель является массивом, объявляется только размерность (число двоеточий), границы массива в описании опущены - они определяются только во время установления связи указателя с объектом-массивом.

Пример

```
REAL, DIMENSION (:, :), POINTER :: X, Y
```

Указатель можно интерпретировать как дескриптор, который содержит всю информацию, необходимую для полного описания и размещения в памяти объекта того же типа с теми же параметрами типа и с той же конфигурацией, что и в объявлении указателя. При объявлении указателя отводится область памяти для дескриптора, но область памяти для самого объекта не отводится. Дескриптор сначала создается пустой и заполняется во время установления связи с объектом.

Связь указателя с объектом может быть установлена:

- при выполнении оператора присваивания указателю с данным указателем в левой части (см. 3.7.2.),
- при выполнении оператора ALLOCATE, когда создается динамический объект (см. 3.7.3.).

После того как установлена связь указателя с объектом, указатель может появиться в любом месте, где может появиться обычный объект того же типа, с теми же параметрами типа и с той же конфигурацией, что и указатель. Значением операнда-указателя является значение объекта, связанного с ним в данный момент.

Аналогичное правило действует и для использования указателя в левой части встроенного оператора присваивания. Для присваивания значения собственно указателю служит уже упоминавшийся оператор присваивания указателю (см. пример в 3.7.2.).

3.7.2. Оператор присваивания указателю устанавливает связь указателя с существующим объектом, оператор NULLIFY разрывает связь указателя с объектом.

Примеры

```
! Атрибут TARGET означает, что к массиву MATR можно обращаться
!  с помощью указателя
REAL, TARGET  :: MATR (M, N)
REAL, POINTER  :: ROW (:), WINDOW (:, :)
ROW => MATR (M, :)       ! ROW связывается со строкой матрицы
! WINDOW связывается с секцией массива MATR
WINDOW => MATR (I-1:I+1, J-1:J+1)
```

Последний пример иллюстрирует еще одно полезное свойство указателя: в тех случаях, когда часто используется одна и та же секция массива, можно не писать каждый раз секцию, а использовать указатель в качестве псевдонима.

3.7.3. Для создания массивов, которые не были заранее явно объявлены в программе, используется оператор ALLOCATE, содержащий указатель. При выполнении такого оператора создается динамический объект, определяются его границы, объекту отводится область памяти, указатель связывается с этим объектом и может в дальнейшем использоваться для ссылки на него. Создаваемые объекты не имеют имени, поэтому ссылаться на них можно только с помощью указателей.

Пример

```
! Объявление  указателей
REAL, DIMENSION (:, :), POINTER  :: A, B
. . .
READ (*, *) N, M

! Создание и размещение динамических массивов
ALLOCATE (A (N, M), B (N, M))
```

При выполнении оператора DEALLOCATE освобождается память от тех динамических объектов, на которые ссылаются указатели, перечисленные в данном операторе.

Этот аппарат работы с создаваемыми массивами целесообразен в тех случаях, когда существенным является использование указателей; в противном случае лучше использовать механизм размещаемых массивов (см. 3.6.).

3.8. Модули

Одним из наиболее важных новых средств языка является модуль - новый вид программных единиц. Программные единицы-модули служат для описания различных объектов (данных, производных типов, определяемых пользователем операций, процедур, интерфейсных блоков, namelist-списков), доступных другим программным единицам. Для доступа к объектам модуля, имеющим атрибут PUBLIC, используется оператор USE. Такие средства описания глобальных объектов являются более общими и гибкими, чем традиционный для Фортрана оператор COMMON.

Использование модулей вместе с аппаратом для описания новых типов и операций является удобным механизмом, обеспечивающим расширяемость языка; это позволяет создавать модули для конкретных приложений. Описание данных и операций над ними в модуле может быть скрыто от пользователя (такие объекты объявляются с атрибутом PRIVATE). Это позволяет при использовании модуля употреблять описанные в нем операции, не вникая в то, как эти операции выполняются (инкапсуляция).

Еще одно преимущество модулей заключается в том, что обеспечивается возможность зависимой (т.е. контролируемой) раздельной компиляции программных единиц. Это объясняется тем что информация, специфицируемая в модуле, доступна процессору при компиляции программных единиц, использующих

данный модуль. Зависимая и в то же время раздельная компиляция повышает надежность программ и позволяет компилятору выполнять более широкий спектр оптимизирующих преобразований.

3.9. Поддержка объектно-ориентированного программирования

Предыдущие разделы уже позволяют читателю убедиться в том, что современный Фортран поддерживает значительную часть методологии объектно-ориентированного программирования (ООП).

ООП – это современная методология программирования. ООП позволяет использовать не только встроенные абстракции (как в традиционном программировании), но и определять собственные абстракции и тем самым позволяет описать программу в терминах близких к прикладной области.

Методология ООП ориентирована прежде всего для написания очень больших программ или набора программ, которые разрабатывают группы программистов. При использовании старых традиционных методов и старых языков программирования, такие программы или системы были трудны для понимания, сопровождения и модификации. Современные языки предлагают новые методы.

В книге [16] обсуждается языковая поддержка ООП для C++ и Ада 95. Язык Фортран в книге отнесен к языкам с низким уровнем абстракции. Это справедливо только в отношении “старого” Фортрана. Что касается современного Фортрана, отметим, что в описании стандарта явно не указывается, какие средства обеспечивают поддержку тех или иных элементов ООП (это и не требуется в официальном документе).

Однако, как показали проведенные исследования (проект РФФИ №00-01-00043), такие средства имеются: расширяемость типов и операций, возможность статического контроля соответствия типов и соответствия формальных и фактических аргументов при вызове процедур, механизм инкапсуляции для реализации абстрактных типов данных, статический полиморфизм и частично наследование.

3.10. Новшества для процедур

Основные новшества, касающиеся процедур, - следующие: внутренние и рекурсивные процедуры, возможность специфицировать аргументы по назначению (IN, OUT, INOUT), возможность использования необязательных и ключевых параметров при вызове процедур, средства задания явного интерфейса, а также

большой набор новых встроенных процедур (в языке более 100 встроенных процедур: математических, справочных и др.).

Явный интерфейс задается с помощью интерфейсного блока; в некоторых случаях он является обязательным, так как позволяет правильно сгенерировать вызов процедуры; в других случаях использование интерфейсного блока позволяет проверить правильность вызовов процедур (например, проверить соответствие типов аргументов).

Важным новшеством в Фортране 95 является атрибут PURE. С помощью этого атрибута пользователь может объявить функцию без побочного эффекта. Вызов такой функции можно использовать в тех случаях, где возможна параллельная обработка без таких нежелательных последствий как недетерминизм.

Новыми являются поэлементные процедуры, т.е. процедуры, аргументами и результатами которых могут быть как скаляры, так и массивы; результат получается таким, как если бы процедура была применена к соответствующим элементам каждого из массивов-аргументов. В Фортране 90 были введены только встроенные поэлементные функции, в Фортране 95 появилась возможность использовать также определяемые пользователем поэлементные процедуры. Поэлементные процедуры обеспечивают дополнительные возможности для выражения параллелизма.

3.11. Средства поддержки параллельности

В предыдущих разделах уже отмечались средства языка, которые могут использоваться для параллельного программирования. Фортран 95 не ориентирован на многопроцессорные системы. Для таких систем разработаны специальные языки, которые будут обсуждаться во второй части обзора. Эти языки существенно используют новые возможности современных стандартов.

В то же время Фортран 90/95 содержит богатый набор средств ориентированных на архитектуру с векторными операциями. Необходимость таких средств (помимо дополнительных удобств) вызвана появлением на современных ЭВМ аппаратных средств векторной обработки.

До появления этих средств в стандарте для таких компьютеров создавались векторизующие компиляторы, помогающие эффективно использовать возможности, заложенные в аппаратуре. Однако автоматическая векторизация (т.е. выявление скрытого параллелизма) требует довольно сложного анализа исходной программы, и при этом некоторые конструкции последовательных яэыков могут затруднить или даже сделать невозможной векторизацию. В некоторые системы программирования

вводились специальные директивы, помогающие компилятору выполнять векторизацию исходной программы, были разработаны и языковые расширения. Достаточно полная информация об исследованиях проблем векторизации содержится в сборнике статей [17] и в статьях [18, 19]; отечественные разработки для ПС-3000 и ЕС1191 описаны в [20, 21].

В современные стандарты, как отмечалось выше, введены средства для работы с массивами и секциями массивов как с целыми объектами, векторные операции, стандартные процедуры для работы с массивами, поэлементные процедуры, условное присваивание массиву под управлением маски, параллельный цикл. Эти средства позволяют компилятору сгенерировать эффективный код с учетом особенностей аппаратуры.

3.12. Поддержка структурного программирования

Современные стандарты языка, в отличие от Фортрана 77, поддерживают структурное программирование. Язык имеет полный комплект современных управляющих структур. В частности, в язык введена структурная конструкция цикла DO - END DO, в которой отсутствует метка. Кроме того, заголовок цикла может иметь как традиционную форму (с управляющей переменной цикла), так и новую форму с ключевым словом WHILE или без управляющей информации.

Введены два новых оператора выхода из цикла EXIT и CYCLE. Оператор EXIT вызывает выход из цикла на оператор, непосредственно следующий за заключительным оператором цикла. Оператор CYCLE вызывает переход на следующую итерацию цикла. Использование операторов EXIT и CYCLE делает программу более наглядной и больше соответствует принципам структурного программирования, чем использование для тех же целей оператора GO TO и меток.

Конструкция выбора SELECT CASE - END SELECT позволяет выбрать для исполнения один из вариантов-блоков операторов в зависимости от значения некоторого выбирающего выражения.

Для удобства написания и чтения программ введены имена конструкций, которые позволяют идентифицировать конструкцию. Имя конструкции не может использоваться для передачи управления. Указание имени конструкции особенно полезно при вложенных конструкциях и является одним из элементов хорошего стиля программирования.

Средства работы с массивами как с целыми объектами (см. 3.5.) также служат улучшению структуры программы, так как во многих случаях позволяют обходиться без циклов и условных операторов.

3.13. Концепция эволюционного развития языка

Представляет интерес принятая концепция эволюционного развития языка. Некоторые конструкции языка Фортран в настоящее время устарели и стали излишними после введения новых элементов; такие конструкции отнесены к категории устаревших черт. Исключение из языка каких-либо элементов может быть весьма болезненным, так как оно может привести к тому, что невозможно будет использовать существующий фонд программного обеспечения на Фортране. Вместе с тем очевидно, что если в язык будут добавляться новые средства без удаления уже ненужных, устаревших элементов, язык станет очень громоздким с большим числом дублирующих элементов.

Учитывая эти противоречивые факторы, было принято следующее решение: элементы языка, которые являются кандидатами на удаление, заранее объявляются устаревшими, нерекомендуемыми для использования.

В работе [9] предложен перечень средств, которые следует признать устаревшими в будущем стандарте Фортрана, даны обоснования и предложения по замене устаревших черт современными элементами языка.

Отметим еще, что современный стандарт требует, чтобы компилятор сообщал пользователю не только конструкции, которые не отвечают требованиям языка, но также и устаревшие конструкции.

4. Перспективы развития языка

В настоящее время разрабатывается проект будущего стандарта, рабочее название которого Фортран 2000 (на некотором этапе использовалось название Фортран 200x). Все средства Фортрана 95 сохранены (кроме быть может признанных устаревшими).

Проект языка Фортран 2000 предусматривает весьма существенные нововведения. Назовем лишь некоторые из наиболее значимых направлений:

- Развитие средств объектно-ориентированного программирования (предполагается полный набор средств ООП).
- Средства взаимодействия с Си.
- Параметризованные производные типы.

- Новые средства ввода/вывода (асинхронный ввод/вывод и др.).
- Новые возможности, касающиеся размещаемых массивов.
- Исключительные ситуации.
- Более полная интеграция с операционной системой.

Для того чтобы эффективно реагировать на новые требования, не дожидаясь принятия следующего стандарта (Фортран 2000), было решено выделить наиболее приоритетные новые черты и выработать проекты для таких новых черт в качестве Технических отчетов. В дальнейшем такие средства должны быть вставлены в неизмененном виде в следующую ревизию основного стандарта.

Эти технические отчеты позволяют реализаторам добавить новые черты в компиляторы Фортрана 95, не дожидаясь завершения разработки будущего стандарта в полном объеме. Если потребуются какие-либо модификации, то разработчики стандарта будут стремиться к тому, чтобы минимизировать изменения в существующих коммерческих реализациях.

К настоящему времени завершена разработка следующих двух Технических отчетов, которые одобрены и уже являются стандартами:

- Новые возможности, касающиеся размещаемых массивов [22].
- Исключительные ситуации для операций с плавающей точкой[23].

Предполагается, что стандарт Фортран 2000 будет утвержден в конце 2004 года. В работе [24] уже содержатся некоторые предложения по дальнейшему развитию языка после утверждения Фортрана 2000.

ЧАСТЬ II
Языки для параллельного программирования, основанные на Фортране

5. Введение. Подходы к реализации параллельности

Появление суперкомпьютеров с параллельной архитектурой, как за рубежом, так и в нашей стране (см., например, [25-27]), приводит к необходимости разработки эффективных и удобных для программиста средств, обеспечивающих возможность реализовать параллельность, которую допускает выбранный им алгоритм.

Достаточно полный обзор и анализ зарубежных и отечественных систем для высокопроизводительных ЭВМ по состоянию на 1990 год содержится в работе [28]. В

работах [29-31] анализируются различные аспекты параллельного программирования. В последующие годы это направление исследований развивалось очень интенсивно.

Некоторые виды распараллеливания могут быть выполнены компилятором автоматически. Проблемам автоматического распараллеливания последовательных программ посвящено большое количество исследований как за рубежом (см., например, [32]), так и в нашей стране (проект РФФИ №97-01-00977, [33]). Описание некоторых зарубежных систем автоматического распараллеливания (наряду с другой весьма полезной информацией) можно найти на сайте http://www.parallel.ru/. В решении этой проблемы имеются определенные результаты, однако автоматическое распараллеливание остается весьма сложной проблемой, а в некоторых случаях решить ее на статическом уровне невозможно.

Разработаны и продолжают разрабатываться как специальные языки для параллельного программирования, так и расширения для традиционных языков в т.ч. и для современного Фортрана [31].

Очевидно, что для обеспечения переносимости с одной платформы на другую необходима стандартизация и унификация таких средств. Унификация средств распараллеливания в языках высокого уровня - довольно сложная проблема. Трудность заключается в том, что имеется большое разнообразие архитектурных решений, позволяющих использовать возможности распараллеливания. Тем не менее в настоящее время уже разработаны системы, включающие средства, которые стали фактическими стандартами для параллельных вычислительных систем различной архитектуры (см. 6-8). Эти средства реализованы для многих параллельных компьютеров.

Рассмотренные в первой части обзора стандарты Фортрана, хотя и содержат некоторые средства поддержки параллельности, не имеют специальных средств, ориентированных на многопроцессорные системы. Однако разработанные на базе этих стандартов параллельные языки существенно используют новые возможности Фортрана 90/95 (динамические и размещаемые массивы, указатели, структуры, модули, явный интерфейс, определяемые пользователем операции, большое число новых встроенных процедур и др.).

Во многих системах параллельность реализуется с помощью специальной библиотеки, содержащей подпрограммы поддержки параллельности; программа на Фортране для таких машин должна содержать обращения к подпрограммам этой библиотеки. Библиотечный подход имеет как преимущества, так и недостатки.

Преимуществом использования библиотек является то, что исходный язык и компилятор остаются неизменными. В то же время использование библиотек менее удобно для разработчиков прикладных программ, чем использование языковых расширений, так как список параметров обычно довольно громоздкий и не соответствует программированию на языке высокого уровня. Так как исходная программа содержит только обращения к библиотечным подпрограммам, то компилятор не может выполнить некоторые виды оптимизации. Кроме того, в некоторых случаях могут быть дополнительные накладные расходы на обращения к подпрограммам библиотек.

Для разработки прикладных программ более удобным и более естественным по сравнению с библиотечными средствами, нам представляется подход, основанный на введении расширений непосредственно в язык. Помимо удобства, этот подход позволяет компилятору выполнить некоторые дополнительные виды оптимизации.

При разработке языковых расширений обычно возникают две основные проблемы:

- выбор специальных средств для поддержки параллельности;
- отображение выбранных средств в конкретную языковую среду.

Сложность и важность первой проблемы обычно не вызывает сомнений, однако и вторая проблема (часто недооцениваемая) весьма нетривиальна, т.к. при внесении расширений необходимо учесть все многообразие уже существующих языковых средств и все тонкости языка.

При разработке языковых средств для поддержки параллельности имеется две основные стратегии:

- создание новых языковых конструкций, или
- разработка специальных директив, которые позволяют пользователю специфицировать некоторые виды параллельности; такие директивы обычно оформляются в виде комментариев и переводятся препроцессором в операторы Фортрана с вызовами подпрограмм, которые и реализуют параллельную обработку.

Директивы-комментарии позволяют программисту использовать один вариант исходной программы и для последовательного, и для параллельного выполнения, поскольку компилятор, ориентированный на последовательное выполнение, будет игнорировать такие комментарии.

Ниже рассматриваются следующие модели параллельных вычислений, для которых разработаны языковые средства:

- модель, ориентированная на архитектуру с распределенной памятью (см. 6);
- модель, ориентированная на архитектуру с общей памятью (см. 7);
- модель, ориентированная на разбиение данных (см. 8).

Средства, ориентированные на архитектуру с векторными операциями, были рассмотрены в первой части данного обзора (см. 3.5. и 3.11.).

6. Средства поддержки параллельности для многопроцессорных систем с распределенной памятью

6.1. Общие сведения

Система с распределенной памятью (distributed memory) обычно состоит из некоторого количества вычислительных узлов, соединенных коммуникационной сетью. Каждый узел имеет только локальную память. Такая архитектура обеспечивает независимое выполнение различных частей параллельной программы на выделенных процессорных узлах. Обмен информацией между различными частями программы осуществляется посредством посылки и приема сообщений по межпроцессорной сети.

Программы для систем с распределенной памятью обычно используют механизм задач (процессов) и передачи сообщений (обмен данными). Исходная программа разбивается на задачи (подзадачи), которые могут выполняться параллельно на разных процессорных узлах, и связь между задачами осуществляется с помощью передачи сообщений. В основе такого подхода лежит фундаментальная модель Хоара [34].

Очевидно, что не все программы позволяют эффективно использовать распределенные системы. Так как обмены сообщениями требуют затрат времени для своего выполнения, важно, чтобы время, необходимое для обмена сообщениями, не доминировало над временем выполнения программы. Распределенные системы целесообразнее использовать для тех вычислительных задач, которые могут быть разбиты на такие слабосвязанные подзадачи, для которых время выполнения существенно больше, чем время обмена между ними.

Имеются разные подходы как к выбору конкретных средств реализации механизма задач и передачи сообщений, так и отображению этих средств в языке. В большинстве систем (PVM, MPI, EXPRESS, P4, PARMACS и др.) механизм задач реализуется с помощью библиотечных вызовов [35-39].

6.2. Система MPI

Из систем, ориентированных на архитектуру с распределенной памятью и представляющих собой набор библиотечных интерфейсов, наибольшее распространение получили MPI и PVM. Система MPI (Message Passing Interface) [37] фактически является международным стандартом, в ней обобщен опыт использования подобных систем, который получил дальнейшее развитие.

MPI включает большой набор средств, среди которых операции передачи сообщений от одного источника (процесса) к другому. Каждому из предусмотренных (по способу синхронизации) видов передачи сообщений соответствует своя операция; необходимая для выполнения операции информация эадается с помощью параметров процедур. Имеется возможность одновременного выполнения вычислений и коммуникаций. Предусмотрены коллективные операции, которые включают как широковещательную передачу сообщений, так и функции редукции. Имеются средства, позволяющие пользователю специфицировать топологию процессов.

Поддерживаются все типы данных, имеющиеся в Фортране и С, имеются и собственные типы данных. Кроме того, для предотвращения коллизии с правилами типов языков верхнего уровня в MPI предусмотрены системные (так называемые "скрытые") объекты, внутреннее представление которых скрыто от пользователя. Предусмотрена возможность конструирования производных типов (структур), которые обеспечивают возможность с помощью одного вызова передать объекты данных разных типов, передать данные, не расположенные в непрерывной области памяти или передать секции массивов.

Для адресации используются группы процессов и коммуникаторы. Коммуникаторы и контексты обеспечивают возможность делить общее коммуникационное пространство на отдельные замкнутые области.

Дальнейшее развитие эти средства получили в системе MPI-2 [39]. MPI не поддерживает объектно-ориентированное программирование, хотя в MPI-2 сделан некоторый шаг в этом направлении. Работа [40] содержит аналитический обзор систем, которые вводят объектную ориентацию в параллельные вычисления с использованием MPI.

В работе [41] описываются средства использования MPI в программах на Фортране 90. Система MPI реализована на многих параллельных компьютерах; система адаптирована для отечественных многопроцессорных вычислительных систем: МВС-100 и МВС-1000 (проекты РФФИ №99-01-00922 и №99-07-90443 [42]).

Подводя некоторые итоги, отметим, что система MPI является хорошим инструментом для распараллеливания широкого класса задач, однако относительно низкий уровень библиотечных средств, чрезмерная детализация приводят к определенным неудобствам при использовании таких систем прикладными специалистами. Подобные системы являются хорошим инструментальным средством для реализации языков более высокого уровня (см. раздел 8).

6.3. Система GNS

В нашей стране разработана и реализована система программирования GNS (включающая Фортран GNS и C GNS; проект РФФИ №96-01-00493) [43-45], которая ориентируется не на библиотечные вызовы, а на введение непосредственно в язык новых конструкций для использования в распределенных системах. Система является развитием предложений в работах [46, 47]. Использование языковых расширений обеспечивает больше удобств для программистов, чем библиотечные вызовы.

Рассмотрим некоторые основные черты этой системы. Прикладная программа может содержать несколько задач, которые могут выполняться параллельно на разных процессорах, при этом число задач не ограничивается числом процессоров.

Каждая задача получает уникальный идентификатор, который служит в качестве адреса отправителя или получателя при передаче сообщения. Задача использует только локальную область памяти без разделения данных с другими задачами.

В системе предусмотрены две модели параллелизма - динамическая и статическая. В статическом варианте все задачи запускаются непосредственно после загрузки. В динамической модели прикладная программа должна содержать одну начальную задачу, в статическом варианте это требование не является обязательным. Динамическая модель предполагает динамическое создание и завершение задач. Выполнение программы начинается с выполнения начальной задачи. Она создает другие задачи, которые, в свою очередь, могут также создавать новые задачи. Динамическое порождение задач позволяет эффективно использовать вычислительные ресурсы и разрабатывать настраиваемые программы.

Обмен информацией между задачами, как в статической, так и в динамической модели, осуществляется с помощью передачи сообщений. Допускается взаимодействие произвольных пар процессоров. Имеются средства

широковещательной передачи сообщений. Предусмотрено три способа (режима) передачи сообщений: синхронный, асинхронный и передача без ожидания.

Для реализации указанных возможностей в язык введены новые операторы (SEND, RECEIVE и др.), новый тип данных для идентификации задач (TASKID), конструкция выбора (SELECT MESSAGE - END SELECT), новый вид программных единиц (программные единицы-задачи) и новые стандартные процедуры.

Существенным фактором является возможность указывать в операторах передачи сообщений произвольный список передаваемых данных, аналогичный списку ввода/вывода языка Фортран, что гораздо удобнее, чем средства, имеющиеся в системах, основанных на библиотечном подходе.

Однако реализованный вариант Фортрана GNS ориентирован на устаревший вариант языка - Фортран 77; это вызвано тем, что базовое программное обеспечение системы, для которой первоначально была предназначена эта разработка, не снабжено компиляторами, соответствующими современным стандартам Фортрана.

Новые средства, введенные в Фортран 90/95, позволяют расширить Фортран GNS. В работе [48] (проект РФФИ № 97-01-00361) описываются предложения по дальнейшему развитию Фортрана GNS, которые ориентированы на новые возможности, предоставляемые современными стандартами Фортрана. Новые средства обеспечивают больше удобств для пользователя и более широкие возможности параллельной обработки.

По сравнению с MPI, языковые средства GNS довольно простые и удобные для прикладного специалиста. Однако они беднее по своим функциональным возможностям. В связи с этим, нам представляется, что работы по развитию Фортран GNS могли бы быть продолжены в следующих направлениях.

- Разработать расширения языковых средств, ориентированных на современные требования, включая объектную ориентацию (некоторые расширения, ориентированные на современные стандарты Фортрана, как отмечалось выше, уже разработаны).
- Реализовать расширенный вариант Фортрана GNS.
- Реализовать GNS следующим образом: исходную программу переводить на Фортран 90/95, дополненный вызовами MPI-процедур. Такой подход используется обычно и для реализации HPF (см. 8.2.), т.е. Фортран 90/95 и MPI являются промежуточными при реализации параллельного языка. (Идея

промежуточного языка была в свое время успешно апробирована при реализации в нашей стране Алгола и Фортрана на АЛМО [49]).

В результате реализации этих предложений пользователь смог бы получить более простой инструмент, чем MPI и более современные средства, чем Фортран GNS, и появилась бы возможность использовать программы, переведенные с Фортрана GNS, на любой системе, имеющей реализацию Фортрана 90/95 и MPI. Последний фактор способствовал бы повышению конкурентноспособности прикладных программ, поскольку они могли бы работать практически на любой вычислительной системе с распределенной памятью.

7. Средства поддержки параллельности для многопроцессорных систем с общей памятью

7.1. Общие сведения

В системах с общей (разделяемой) памятью (shared memory) несколько физических процессоров могут совместно использовать общую область памяти. Проблемы, которые при этом возникают, решаются с помощью синхронизации процессов.

Для вычислительных машин с общей памятью, так же, как и для других видов архитектуры, разработаны системы, обеспечивающие автоматическое распараллеливание (в тех случаях, где это возможно), системы, в которых пользователь вставляет обращения к библиотечным процедурам, или компилятор распараллеливает обычный последовательный код, используя директивы-комментарии. В нашей стране также были разработаны средства спецификации параллельности для многопроцессорного (с общей памятью) варианта ЕС1191 [21].

Работы по унификации расширений языков средствами спецификации параллелизма выполнялись первоначально группой PCF (Parallel Computing Forum) [50], в дальнейшем X3H5 ANSI [51]. Эти группы объединили представителей крупнейших фирм, которые накопили большой опыт использования многопроцессорных систем с общей памятью. Разработана языково-независимая модель параллелизма и расширения Фортрана на основе этой модели.

Проект X3H5 положен в основу стандарта OpenMP [52, 53]. Спецификации для Фортрана были утверждены в 1997 году, для C/C++ в 1998 году. Работа по дальнейшему развитию OpenMP продолжается [52].

7.2. Модель параллелизма

Концептуальная модель параллелизма, заложенная в X3H5 и OpenMP, очень схожая, хотя имеются некоторые различия в конкретной реализации модели. Ниже кратко описываются основные принципы этой модели.

Программа начинает выполняться как один процесс, который называется главной нитью. Этот процесс выполняется последовательно до тех пор, пока не дойдет до первой директивы, специфицирующей начало параллельной области. В этот момент порождается бригада нитей (процессов). Порождающая (главная) нить является также членом этой бригады. Число нитей в бригаде задается по-разному, например, с помощью переменной окружения или путем вызова соответствующей подпрограммы.

Фрагмент программы внутри параллельной области выполняется всеми нитями бригады на разных процессорах. Главная нить ждет завершения выполнения всех нитей и продолжает выполнение только одна. Такие бригады могут образовываться не один раз.

Внутри параллельной области могут находиться параллельные конструкции (параллельный цикл, параллельные секции), которые определяют распределение (разделение) работы между членами бригады, но которые не создают новых нитей. Допускается вложенность параллельных областей и параллельных конструкций.

Для реализации этой модели в язык вводятся директивы, специфицирующие параллельные области и параллельные конструкции, введены средства синхронизации, набор процедур run-time поддержки и переменные среды.

Схематически эта модель может быть представлена следующим образом:

```
program
! Активизируется начальный процесс

! Последовательное выполнение
parallel            ! Начало параллельной области

! Каждый член бригады выполняет одни и те же действия
  pdo i=1, n                ! Начало конструкции "параллельный цикл"

! Итерации распределяются среди членов бригады
. . .
  end pdo           ! Конец конструкции "параллельный цикл"
. . .
```

```
  psection          ! Начало конструкции "параллельные секции"
! Действия выполняются всеми членами бригады
. . .
    section         ! Параллельная секция
. . .
    section         ! Параллельная секция
. . .
  end psection              ! Конец конструкции "параллельные секции"
. . .
end parallel                ! Конец параллельной области
! Последовательное выполнение
! Другие параллельные конструкции
. . .
end
```

7.3. Спецификации для Фортрана

В OpenMP для Фортрана директивы оформляются в виде комментария, начинающегося с символов !$OMP, *$OMP или C$OMP. В X3H5 директивы имеют вид обычных операторов Фортрана (хотя допускается их оформление и в виде комментария), поэтому выбор ключевых слов для таких операторов делался с учетом того, чтобы не было коллизии с ключевыми словами стандартного Фортрана. В Open MP такого принципа не придерживались, коллизии не произойдет, поскольку директивы оформляются как комментарии, но тем не менее нам представляется, что и в этом случае было бы целесообразнее использовать непересекающиеся имена, тем более, что набор таких ключевых слов был уже подготовлен в проекте X3H5. Ниже рассматриваются основные конструкции этих систем (для сокращения префикс комментария опускается).

Конструкция, задающая параллельную область, имеет вид:

```
PARALLEL [список-спецификаций]
. . .
END PARALLEL
```

В заголовке в списке спецификаций могут описываться классы данных, условие выполнения и другие дополнительные опции (они несколько отличаются в рассматриваемых системах).

Объекты данных, используемые в таких параллельных конструкциях, могут быть приватными для каждой нити или общими, т.е. разделяемыми между нитями бригады. Счетчики циклов всегда приватные. Имеются и другие классы переменных.

Параллельные конструкции, которые определяют разделение работы между членами бригады, задаются с помощью следующих директив:

Параллельный цикл:

DO-END DO (OpenMP)

PDO-END PDO (X3H5)

Параллельные секции:

SECTIONS - SECTION - END SECTIONS (OpenMP)

PSECTIONS - SECTION - END PSECTIONS (X3H5)

Параллельный цикл определяет распределение итераций между нитями. Конструкция параллельные секции позволяет нескольким нитям выполнить параллельно участок программы, разбитый на независимые секции.

Если внутри параллельной области имеется только одна параллельная конструкция, то допускается укороченная запись заголовка:

PARALLEL DO - END PARALLEL DO

PARALLEL SECTIONS - END PARALLEL SECTIONS (OpenMP)

PARALLEL PDO - END PARALLEL PDO

PARALLEL PSECTIONS - END PARALLEL PSECTIONS (X3H5)

В заголовках всех перечисленных выше конструкций могут быть дополнительные спецификации (класс переменных, дисциплина распределения работы между нитями, способ планирования последовательности выполнения и др.).

Кроме директив, определяющих распределение работ, внутри параллельной области допускается использовать конструкцию SINGLE - END SINGLE, которая указывает, что входящий в нее блок программы выполняется только один раз.

В системе предусматривается неявная синхронизация, которая может автоматически выполняться системой реализации, и средства явной синхронизации, задаваемые пользователем (барьеры, критические области и др.); здесь имеются различия в рассматриваемых системах.

Из встроенных функций отметим следующие: функции, устанавливающие число нитей в бригаде, функции, возвращающие число нитей, возвращающие номер текущей нити в бригаде и др.

8. Средства параллельности, ориентированные на разбиение данных

8.1. Общие сведения

Одним из актуальных направлений в создании языковых средств поддержки параллельности является подход, ориентированный на разбиение данных. Суть этого похода заключается в следующем. В язык вводятся средства, позволяющие пользователю специфицировать разбиение массивов данных и распределить их по процессорам.

Разбиение данных производится исходя из соображения, что операции над двумя или более объектами выполняются быстрее, если они находятся на одном и том же процессоре, и на разных процессорах операции можно выполнить параллельно.

При таком подходе каждый процессор содержит только подмножество распределенных данных; для доступа к данным, хранящимся в другом процессоре, компилятор генерирует операции, обеспечивающие межпроцессорную коммуникацию, которая необходима для такого доступа. При этом от пользователя не требуется явно специфицировать способ синхронизации. Такие средства могут быть использованы для параллельных компьютеров различной архитектуры.

Соответствующие расширения для Фортрана были первоначально разработаны и реализованы в двух проектах: Fortran D (Техасский университет) [54] и Fortran Vienna (Венский университет) [55]. Хотя на концептуальном уровне в этих подходах много общего, языки Fortran D и Fortran Vienna отличаются друг от друга.

8.2. Язык HPF

Дальнейшее развитие этот подход получил в языке HPF (High Performance Fortran) [56], который является расширением Фортрана 90, существенно использует его новые возможности и фактически является международным стандартом.

HPF содержит директивы для описания способов разбиения данных между параллельно работающими процессорами и некоторые средства для явного указания параллельности.

HPF – язык высокого уровня. На программиста возлагается в основном ответственность за распределение данных, а представление программы в виде системы взаимодействующих процессов осуществляется компилятором.

Более новая версия этого языка HPF 2.0 [57] основывается на Фортране 95 и содержит ряд нововведений. В HPF 2.0 внесены серьезные расширения по сравнению с первой редакцией. Поскольку HPF 2.0 является расширением Фортрана 95, все новые черты, которые включены в Фортран 95, неявно включены и в HPF 2.0. Имеются также существенные собственные новшества. Далее мы будем говорить об HPF, имея в виду и первую и вторую редакции.

Директивы HPF имеют вид комментария, который начинается с символов !HPF$, *HPF$ или CHPF$. Для иллюстрации рассмотрим назначение некоторых из этих директив.

Директива DISTRIBUTE определяет способ распределения данных по абстрактным процессорам. Специфицируется характер распределения (блочный, циклический, блочно-циклический) для массива или для каждого измерения. Отображение абстрактных процессоров на физические выполняет система реализации.

Директива ALIGN специфицирует совместное размещение данных, т.е. указываются массивы или их части, которые должны быть распределены на одном и том же абстрактном процессоре; таким образом, директива позволяет специфицировать взаимное выравнивание массивов друг с другом как внутри, так и между размерностями в зависимости от характера вычислений. Эта информация позволяет компилятору минимизировать количество пересылок между процессорами.

В качестве средств явного указания параллельности в первую версию HPF был включен оператор и конструкция FORALL (см. 3.5.). Из описания HPF 2.0 удалено описание FORALL, так как эти средства теперь имеются в основном варианте стандарта Фортрана – в Фортране 95. Директива INDEPENDENT указывает компилятору, что итерации в следующем DO-цикле и операции в следующем FORALL могут выполняться независимо, в любом порядке. Кроме того, для поддержки параллельности в язык добавлены некоторые встроенные функции, включая и справочные. В HPF2 включены некоторые дополнительные средства для спецификации параллельных вычислений.

Поскольку HPF является языком высокого уровня, существуют некоторые операции, которые трудно или невозможно выразить на этом языке. EXTRINSIC

процедуры позволяют описать явный интерфейс с процедурами, написанными с использованием других средств, например, различные библиотеки явной поддержки механизма передачи сообщений.

В настоящее время на многих параллельных компьютерах имеются реализации HPF. Компиляторы HPF обычно осуществляют перевод HPF-программ на Фортран 90/95, дополненный вызовами подпрограмм одной из библиотечных систем : MPI, PVM и др.

8.3. Система DVM

Средства распределения данных в отечественной системе DVM (Фортран DVM и C DVM; проекты РФФИ №96-01-1745 и № 99-01-00209 [58-60]) основаны на аналогичных средствах, имеющихся в HPF. В отличие от HPF, в системе DVM имеются дополнительные средства для распределения вычислений, средства спецификации удаленных данных и средства, специфицирующие параллелизм задач.

Такой подход позволяет пользователю лучше учесть особенности архитектуры и создать более эффективную программу, но в то же время возлагает на программиста дополнительную работу, т.е. усложняет программирование. Базовым языком для Фортрана DVM является Фортран 77.

Для обеспечения совместимости с HPF (который фактически является международным стандартом), по нашему мнению, было бы целесообразно развитие языка Фортран DVM в следующем направлении.

- В качестве базового языка целесообразно выбрать современный стандарт Фортран 90/95, который реализован на разных платформах практически для всех современных компьютеров.
- Все средства, которые имеются и в HPF (или в его стандартном подмножестве), и в DVM, должны полностью совпадать, включая и префикс комментария (полная синтаксическая и семантическая идентичность).
- Те средства DVM, которые отсутствуют в HPF, могут иметь собственный префикс комментария (например, CDVM$).

В системе HPF-DVM сделан некоторый шаг для обеспечения совместимости с HPF, но и в этом случае нет полной идентичности со стандартным подмножеством HPF (которое определено в описании языка), и, кроме того, ориентация делается на Фортран 77, а не на подмножество Фортрана 90.

Реализация указанных предложений позволила бы написанные на языке HPF программы выполнять с помощью средств системы DVM и, наоборот, программы, написанные на Фортране DVM, смогут выполняться на компьютерах, для которых реализован HPF, при этом HPF-компилятор будет просто игнорировать дополнительные средства Фортрана DVM. Это повысило бы мобильность создаваемых прикладных программ, и при этом пользователь мог бы делать выбор между эффективностью, с одной стороны, и более простым программированием - с другой. Это способствовало бы повышению конкурентноспособности прикладных программ, поскольку они могли бы работать практически на любой вычислительной системе.

9. Co-array Fortran

Представляется интересным проект для параллельного программирования - Co-array Fortran [61] (более раннее название F--). Язык является небольшим расширением Фортрана 95.

Co-array Fortran реализует модель SPMD. Программа на этом языке интерпретируется так, как если бы она была неоднократно сдублирована, и все копии (экземпляры) выполнялись асинхронно; каждый экземпляр имеет свой собственный набор объектов данных.

Для доступа к данным расширен синтаксис массивов Фортрана 95. Добавлен новый объект, называемый “co-array”. В объявлении такого объекта добавляются квадратные скобки. Например, объявление

```
real, dimension (n) [*] :: x, y
```

означает, что каждый экземпляр программы имеет два вещественных массива x и y размером n.

Ссылка на массив без квадратных скобок относится к локальным массивам. Квадратные скобки используются для доступа к массивам из другого экземпляра; там, где есть квадратные скобки, включается коммуникация.

Например, если оператор

```
x(:) = y (:) [m]
```

выполняется на всех экземплярах, и при этом "m" имеет одинаковые значения на каждом из них, результат такой, что каждый экземпляр программы выбирает массив "y" из экземпляра "m" и помещает его в локальный массив "x".

Для синхронизации в языке имеются встроенные процедуры; имеются также встроенные функции, возвращающие количество экземпляров и номер текущего экземпляра.

Язык Co-array Fortran – довольно простой; концепции языка ближе к моделям передачи сообщений, чем к моделям, ориентированным на разбиение данных, но предлагаемые средства более высокого уровня, чем в системах передачи сообщений, так как здесь нет необходимости в дополнительной пересылке данных.

10. Система Норма

Рассмотренные выше подходы к описанию параллельности основаны на введение расширений в Фортран. Имеются и другие подходы. Так, в нашей стране разработана и реализована система Норма (проекты РФФИ №95-01-00575-а , №98-01-00987, №99-01-00842, №01-01-00411, №02-01-01114 [62-64]). Язык системы является специализированным, он позволяет описывать широкий класс задач математической физики в терминах привычных для прикладного специалиста и, кроме того, содержит средства интерфейса с Фортраном.

Система позволяет получить выходную программу на языке Фортран, дополненную различными средствами распараллеливания для ЭВМ с различной параллельной архитектурой. Кроме того, Норма может обеспечить интерфейс с Фортраном. Это позволяет записать часть программы на более универсальном языке, каким является Фортран.

Описание на языке Норма обеспечивает возможность выявления естественного параллелизма. Прикладному специалисту при использовании системы Норма требуется меньше информации о конкретной архитектуре, компиляторе, организации библиотек и т.п., и при этом он может полностью использовать все возможности используемой вычислительной системы.

11. Сравнение систем

Механизм задач и передачи сообщений, используемый для распределенных систем, и средства синхронизации, используемые в системах с общей памятью, позволяют лучше учесть особенности архитектуры и более эффективно распараллелить программу. Однако эти средства, по нашему мнению, могут быть сложными для разработчиков прикладных программ.

В то же время примитивы нижнего уровня являются хорошим инструментальным средством для реализации параллельных языков высокого

уровня. В частности, компиляторы HPF обычно осуществляют перевод HPF-программ на Фортран 90/95, дополненный вызовами подпрограмм одной или нескольких библиотечных систем : MPI, PVM, PARMACS и др.

Средства, предлагаемые в языке HPF, обеспечивают более высокий уровень программирования. Такие средства более простые и более привычные для разработчиков прикладных программ, они не требуют от пользователя детального знания проблем, возникающих при распараллеливании программ, и избавляют его от необходимости заботиться об организации параллельных процессов (задач) и средствах их взаимодействия.

Однако программы, написанные на языке более высокого уровня, могут оказаться менее эффективными при работе на конкретном компьютере. Эффективность зависит от алгоритма, используемого в прикладной программе, конкретной архитектуры и компилятора.

Некоторые компромиссные решения предлагаются в системах, описанных выше. Учитывая противоречивые факторы, программист может сделать выбор между простым и удобным программированием и эффективностью.

12. Реализации современного Фортрана

Разработкой и поддержкой компиляторов для различных видов компьютеров занимаются многие ведущие разработчики системного программного обеспечения. Современные стандарты Фортрана реализованы практически для всех вычислительных систем и могут использоваться на различных платформах. Реализованы также расширения стандартов для параллельной обработки: HPF, OpenMP, Co-Array Fortran и др.

Разработчики компиляторов предлагают современный набор инструментов (включая средства отладки, средства визуализации и др.) для разработки Фортран-приложений. В некоторых компиляторах уже реализованы средства, которые войдут в следующий стандарт (Фортран 2000). Это те средства, которые описаны в Технических отчетах, ставших стандартами (см. 4.). Созданы и другие программные продукты, связанные с современным Фортраном (различные анализаторы, конверторы, переводящие прикладную программу с Фортрана 77 на современный Фортран , разнообразные системы тестов, графические библиотеки и др.).

Кроме того, разработаны математические библиотеки, в том числе и для параллельной архитектуры (IMSL).

13. Заключение

В заключение хотелось бы остановиться еще на одной проблеме, имеющей непосредственное отношение к исследованиям, которые обсуждались в обзоре.

В последнее время известные ученые нашей страны обратили внимание на то, что в России ‘’индустрия программного обеспечения” занимает слишком скромное место (см., например, [65, 66]), несмотря на то, что в стране имеются все предпосылки для создания и развития этой “экспортной отрасли”. Причины такого положения дел и пути выхода из этой ситуации неоднократно отмечались в печати.

Мы хотели бы обратить внимание еще на одну причину, которая в нашей стране не всегда принимается во внимание. Для того чтобы прикладная программа стала программным продуктом и была конкурентноспособной на зарубежном рынке, она должна быть написана на языке, который является международным стандартом, с использованием современных технологий, предоставляемых этим стандартом. К сожалению, этим вопросам не уделяется должного внимания в курсах программирования в высших учебных заведениях нашей страны [67].

Если говорить о Фортране, которому посвящен данный обзор, то старые варианты языка, которые еще часто используются в нашей стране, не позволяют применять современные технологии. Необходимость использования именно стандартных средств языка обосновывается в разделе 2.1.

Для развития фундаментальной науки, безусловно, надо проводить исследования, разрабатывать и реализовывать новые языковые средства, которые в дальнейшем могут быть предложены международному сообществу.

Однако для прикладных программ предпочтение следует отдавать современным стандартным средствам.

Это, по нашему мнению, позволит выйти на рынок не только с разработанными в нашей стране вычислительными алгоритмами и методиками расчетов (которые высоко ценятся за рубежом), но и с программным продуктом.

Литература

1. ISO/IEC 1539-1: 1997 Information technology - Programming languages - Fortran - Part 1: Base Language
2. ISO/IEC 1539: 1991(E) Information technology - Programming languages - Fortran
3. Adams J., Brainerd W., Martin J., Smith B., Wagener J. Fortran 90 . Handbook. (Complete ANSI/ISO Reference) – Intertext Publications. McGraw Hill, 1992.
4. Фортран 90. Международный стандарт. Перевод с англ. С.Г.Дробышевич, редактор перевода А.М.Горелик. М.: Финансы и статистика, 1998
5. Горелик А.М., Ушкова В.Л. Фортран сегодня и завтра. - М.: Наука, 1990
6. Меткалф М., Рид Дж. Описание языка программирования Фортран 90. Перевод с англ. П.А.Горбунова М.: Мир, 1995
7. Горелик А.М. Современные международные стандарты языка Фортран. //Программирование, 2001, №6. (English translation: Gorelik A.M. Up-to-Date International Standards of the Fortran Programming Language. //Programming and Computer Software, vol.27, №6, 2001).
8. Горелик А.М. Фортран жил, жив и будет жить. //Открытые системы сегодня, №1, 1995
9. Горелик А.М. Эволюция языка Фортран. Устаревшие черты и современные элементы языка для их замены. - Препринт ИПМ РАН, М., 1997, №66
10. Горелик А.М. Современный Фортран. Состояние и перспективы развития. - Препринт ИПМ РАН. М., 1998, №71
11. Горелик А.М. На пути к Фортрану 200х. - Препринт ИПМ РАН, М.,1999, №64
12. Горелик А.М. Средства поддержки параллельности в современном Фортране. Препринт ИПМ РАН, М., 1999, №75
13. Горелик А.М. Средства поддержки мобильности и надежности программ в современном Фортране. Препринт ИПМ РАН, 2000, №55.
14. ISO/IEC 1539-2 : 2000(E): Information technology - Programming languages - Fortran Part 2 of the Fortran Standard (Varying length strings).
15. ISO/IEC 1539-3 : 1997 Information technology - Programming languages - Fortran - Part 3 of the Fortran Standard (Conditional compilation).
16. Бен-Ари М. Языки программирования. Практический сравнительный анализ. Пер. с англ. под ред. Штаркмана В.С. - М.: Мир, 2000
17. Векторизация программ: теория, методы, реализация. Сб. статей. Пер. с англ. и нем. под ред. Чинина Г.Д. - М.: Мир, 1991

18. Luecke G., Haque W., Hoextra J., Jespersen, Coyle J. Evaluation of Fortran Vector Compilers and Preprocessors. Software – Practice and Experience, vol. 21(9), 1991 pp.891-905
19. Ina H., Kamiya S., Mikami J. Languages and Software Development Tools for supercomputers. // Computer Phisics Communications. 38, 1985, 211-219.
20. Задыхайло И.Б., Зеленецкий С.Д., Платонова Л.Н., Поддерюгина Н.В., Седова И.М., Эйсымонт Л.К. ФОРА – ЕС: Система программирования Фортран IV для многопроцессорного вычислительного комплекса ПС-3000. - Препринт ИПМ АН СССР, М., 1987, №17.
21. Платонова Л.Н., Горелик А.М., Задыхайло И.Б., Зеленецкий С.Д., Поддерюгина Н.В. Расширение языка Фортран для супер-ЭВМ. - Сб. Проблемы повышения эффективности использования ЭВМ большой производительности. //ВЦ АН СССР. М.:1989
22. TR 15580 ISO/IEC JTC1/WG5 №1372. Floating point exception handling.
23. TR 15581 ISO/IEC JTC1/WG5 №1374. Enhanced data type facilities.
24. Nagle D. Next Standard. ISO/IEC JTC1/WG5 №1460
25. Фортов В.Е., Савин Г.И., Левин В.К., Забродин А.В., Шабанов Б.М. Создание и применение системы высокопроизводительных вычислений на базе высокоскоростных сетевых технологий. //Информационные технологии и вычислительные системы. М.: 2002, №1.
26. Забродин А.В. Параллельные вычислительные технологии. Состояние и перспективы. - Препринт ИПМ РАН, М., 1999, №71.
27. Кузьминский М., Волков Д. Современные суперкомпьютеры: состояние и перспективы. // Открытые системы, 1995, №6.
28. Итоги науки и техники. Вычислительные науки, т.3. Черняев А.П. Системы программирования для высокопроизводительных ЭВМ. – ВИНИТИ, М., 1990
29. Задыхайло И.Б., Зеленецкий С.Д. Механизмы синхронизации в языках параллельного программирования. – Известия АН СССР, 1986, №5, 129-174.
30. Горелик А.М., Задыхайло И.Б. Расширения Фортрана для многопроцессорных систем с распределенной памятью. - М., Препринт ИПМ РАН, 1992, №55
31. Горелик А.М. Средства поддержки параллельности в языках программирования. //Открытые системы, 1995, №2
32. Zhiyu S., Zhiyuan L., Pen-Chung Y. An emperical study of Fortran programs for parallelizing compilers. // IEEE Trans. on Parallel and Distributed Systems, July 1990.

33. Воеводин В.В. Отображение проблем вычислительной математики на архитектуру вычислительных систем. // Вычислительные методы и программирование. Изд. МГУ, 2000, 105-112
34. Хоар Ч. Взаимодействующие последовательные процессы. М.: Мир 1989
35. Calkin R., Hempel R., Hoppe H.-C, Wypior P. Portable Programming with the PARMACS message passing library. //Parallel Computing, № 20, 1994.
36. Geist Al., Beguelin A., Dongarra J., Jiang W., Manchek R., Sunderam V. PVM: Parallel Virtual Machine - A Users' Guide and Tutorial for Networked //Parallel Computing, 1994
37. Document for a Message-Passing Interface. Message Passing Interface Forum. May 5, 1994
38. Воеводин Вл.В. Технология параллельного программирования. Message Passing Interface. (htpp://parallel.srcc.msu.su/vvv/mpi.html).
39. MPI Forum. MPI-2: extensions to the Message-Passing Interface. 1998 (htpp://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html).
40. Рычков В.Н., Красноперов И.В., Копысов С.П. Промежуточное программное обеспечение для высокопроизводительных вычислений. // Научный журнал. Вычислительные методы и программирование. Изд. МГУ , 2001, том 2, №2, 117-132.
41. Hennecke M. A Fortran interface to MPI version 1.1 (http://www.uni-karlsruhe.de/"Michael.Hennecke/)
42. Абрамова В.А., Вершубский В.Ю., Поздняков Л.А., Храмцов М.Ю., Шеина Н.П. Система программирования MPI-MVC. Технология подготовки прикладной программы в среде MS-DOS. - Препринт ИПМ РАН №36, М., 1999
43. Абрамова В.А., Вершубский В.Ю., Горелик А.М., Лапыгин А.Е., Поздняков Л.А., Титова Т.И., Фисун В.А., Храмцов М.Ю., Шеина Н.П. Система программирования GNS. Описание языка Фортран GNS. - Препринт ИПМ РАН N 59, М., 1997 (English translation: Abramova V.A., Fisun V.A., Gorelik A.M., Khramtsov M.Yu., Pozdnyakov L.A., Sheina N.P., Titova T.I., Vershubskii V.Yu. GNS Programming System. Fortran GNS Language Description. //The preprint of the KIAM RAS, 1998, №28.
44. Абрамова В.А., Вершубский В.Ю., Горелик А.М., Лапыгин А.Е., Поздняков Л.А., Титова Т.И., Фисун В.А., Храмцов М.Ю.,Шеина Н.П. Система программирования GNS. Описание языка Си GNS. Препринт ИПМ РАН N 64, М., 1997 (English translation: Abramova V.A., Fisun V.A., Gorelik A.M., Khramtsov M.Yu.,

Pozdnyakov L.A., Sheina N.P., Titova T.I., Vershubskii V.Yu. GNS Programming System. C GNS Language Description. //The preprint of the KIAM RAS, 1998, №27.

45. Абрамова В.А., Вершубский В.Ю., Поздняков Л.А., Храмцов М.Ю., Шеина Н.П. Система программирования GNS. Технология подготовки прикладной программы в среде UNIX для вычислительного комплекса МВС-1000. - Препринт ИПМ РАН №20, М., 2000
46. Thole C.-A. Proposal for a Fortran Syntax Specification for distributed memory architectures. Version 1.0, April 1990, SUPRENUM GmbH, Bonn, 1990.
47. Solchenbach K. Suprenum – Fortran an MIMD/SIMD language. // Supercomputer, vol.6, №2, 1989
48. Горелик А.М. Фортран для многопроцессорных систем с распределенной памятью. Расширения языка Фортран GNS. - Препринт ИПМ РАН №26, М., 1998
49. Камынин С.С., Любимский Э.З. Алгоритмический машинно- независимый язык АЛМО. // Алгоритмы и алгоритмические языки. Вып.1. М. ВЦ АНСССР, 1968
50. The Parallel Computung Forum presents. PCF Fortran, v.3.0, Apr.,1990
51. X3H5 Parallel Extensions for Fortran. Apr., 1993 Document Number X3H5/93-SD2-Revision A.
52. Open MP. Simple, Portable, Scalable SMP Programming. (htpp://www.openmp.org/)
53. Кузьминский М. OpenMP: средства распараллеливания для многопроцессорных систем. //Открытые системы, 1998, №3.
54. Hiranandani S., Kennedy K., Tseng C.-W. Fortran D for MIMD Distributed-Memory Machines. Comm. of the ACM, vol. 35, N 8 (August 1992), 66-80
55. Zima H., Brezany P., Chapman B., Mexrotra P., Schwald A.Vienna Fortran. A Language Specification. Version 1.1
56. High Performance Fortran Forum, Language Specification, v.1., 1993
57. ftp://softlib/rice.edu/pub/HPF/hpf-v20.ps.gz
58. Коновалов Н.А., Крюков Н.А., Михайлов С.Н., Погребцов А.А. Fortran DVM – язык для разработки мобильных параллельных программ // Программирование 1995, №1, 49-54
59. Крюков В.А., Р.В.Удовиченко Р.В., Отладка DVM-программ. //Программирование №12, 2001, 19-29
60. Система DVM. htpp://www.keldysh.ru/dvm
61. Numrich W., Reid J. Co-array Fortran for parallel programming. //ISO/IEC JTC1/SC22/WG5 №1317

62. Андрианов А.Н., Бугеря А.Б., Ефимкин К.Н., Задыхайло И.Б. Норма. Описание языка. Рабочий стандарт. - Препринт ИПМ РАН №120, 1995.

63. Задыхайло И.Б, Ефимкин К.Н. Содержательные обозначения и языки нового поколения. //Информационные технологии и вычислительные системы. 1996, №2, 46-58

64. Андрианов А.Н. Система Норма. Разработка, реализация и использование для решения задач математической физики на параллельных ЭВМ. – Дисс. на соискание ученой степени доктора физико-математических наук. М., 2001.

65. Фортов В.Е. Индустрия программного обеспечения – это шанс для России. // Известия, 2000, 22 ноября.

66. Фортов В.Е. Обустроить в России Силиконовую долину. // Известия, 2002, 15 марта.

67. Горелик А.М. О целесообразности изучения современного Фортрана в вузах. //Программирование, 1996, №3. (English translation in Programming and Computer Software, 1996, №6).

Перечень публикаций, появившихся после завершения работы по обзору.

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. - Санкт-Петербург: БХВ-Петербург, 2002.

2. Горелик А.М. Объектно-ориентированное программирование на современном Фортране. - Препринт Института прикладной математики им. М.В.Келдыша РАН, 2002, №70.

3. Немнюгин С.А., Стесик О.Л. Параллельное программирование для многопроцессорных вычислительных систем. - Санкт-Петербург: БХВ-Петербург, 2002.

Б. Совокупность исследований, финансируемых РФФИ

В обзоре рассмотрены два основных комплекса проблем:

1. Развитие языка Фортран с учетом развития архитектуры вычислительных систем и технологии программирования. Разработка новых международных стандартов языка и поддержка действующих стандартов. Разработка методологии использования новых средств языка.

2. Разработка, развитие и реализация средств спецификации параллельности на базе современного Фортрана для решения больших задач вычислительного характера на ЭВМ с различной параллельной архитектурой.

Б.1. В рамках решения первого круга проблем при финансовой поддержке РФФИ (проекты №97-01-00361 и №00-01-00043) проведены исследования действующих и разрабатываемых современных международных стандартов языка Фортран и сравнение их с более ранними, но еще используемыми в нашей стране версиями языка. На основе проведенного анализа разработаны некоторые предложения по усовершенствованию языковых средств при очередной ревизии Фортрана. Кроме того, разработаны рекомендации для прикладных специалистов по эффективному использованию новых средств современного Фортрана.

Наибольшее внимание уделено следующим проблемам.

- Выявление средств, обеспечивающих создание более мобильных и более надежных прикладных программ.
- Анализ средств поддержки объектно-ориентированного программирования.
- Исследования, связанные с введением в язык средств периода компиляции.
- Выявление некоторых нерегулярностей в языке и разработка предложений, позволяющих их устранить.
- Исследования, связанные с развитием средств языка, которые обеспечивают удобный интерфейс Фортран-программ с С-программами.
- Анализ средств параллельности в стандарте.
- Разработка предложений по перечню средств, которые надо признать устаревшими в Фортране 2000; обоснования и предложения по их замене современными элементами языка.

Результаты проведенных исследований используются и могут быть использованы в дальнейшем в различных областях научных исследований, при решениии задач, требующих большого объема вычислений, а также для обучения современным технологиям программирования с использованием Фортрана.

Б2. В рамках решения второго круга проблем при финансовой поддержке РФФИ выполнены следующие исследования.

1. Развитие средств спецификации параллельности для распределенных систем.

В рамках проектов №96-01-00493 и №99-07-90443 разработаны и реализованы языки Фортран GNS и C GNS. Разработаны конкретные предложения по дальнейшему развитию Фортрана GNS на основе современных международных стандартов Фортрана (проект №97-01-00361). Система MPI адаптирована для отечественного многопроцессорного вычислительного комплекса МВС-100 и МВС 1000 (проект №99-01-00922). Результаты использованы для решения ряда задач математической физики.

2. Развитие средств спецификации параллельности, ориентированных на разбиение данных.

Этому направлению посвящены проекты №96-01-1745, № 99-01-00209 и №02-01-00752. Разработана и реализована для различных платформ система DVM, включающая Фортран DVM и C DVM. Система используется для решения задач математической физики.

3. Исследование средств спецификации параллельности для архитектуры с общей памятью. Такие исследования выполняются в проекте № 98-07-90290.

4. Непроцедурные языки для спецификации параллельности.

Этому направлению посвящены проекты №95-01-00575-а , №98-01-00987, №99-01-00842, №01-01-00411 и №02-01-01114. Разработан и реализован непроцедурный специализированный язык и система Норма. Система использована для решения (на ЭВМ различной параллельной архитектуры) ряда практических задач математической физики на статических структурных, неструктурных и адаптивных вложенных сетках.

5. Исследование методов решения задач автоматической генерации программ для современных компьютеров (проект №97-01-00977). Разработаны и апробированы различные варианты генерации параллельных программ по их последовательным аналогам, написанным на Фортране 77.

6. Организация всероссийского информационно-аналитического центра по параллельным вычислениям в сети Интернет .

Этому направлению посвящен проект №99-07-90230. Создан сайт www.parallel.ru, который широко используется, регулярно пополняется новой

полезной информацией. В частности, сайт содержит информацию по средствам спецификации параллельности в Фортране.

В. Степень взаимного соответствия проблематики проектов РФФИ и проблем, указанных в разделах А и Б.

Степень взаимного соответствия рассматривается для каждого из двух основных комплексов проблем, отмеченных выше.

В.1. По первому кругу проблем.

Современные международные стандарты, как уже отмечалось, являются результатом совместной деятельности экспертов многих стран. Специалисты нашей страны неоднократно участвовали в обсуждениях, голосованиях, посылали свои предложения, которые опубликованы в документах международной рабочей группы; многие предложения приняты во внимание; вклад отечественных специалистов неоднократно отмечался в официальных документах. Так, в итоговой резолюции встречи рабочей группы (ISO/IEC JTC1/SC22/WG5 №1412) предложения России, направленные по электронной почте, одобрены и отмечены благодарностью. В отчете 22-му подкомитету (ISO/IEC JTC1/SC22/WG5 №1221) координатор рабочей группы M. Ellis отметил Россию как страну, которая принимает активное участие в дискуссиях по электронной почте, хотя не имеет возможности участвовать во встречах.

Тем не менее, нельзя сказать, что наши предложения оказали существенное влияние на развитие данного направления. Хотя проекты РФФИ (указанные в Б.1.) выполнены с учетом деятельности международной рабочей группы, непосредственное участие в рабочей группе не поддерживается (отсутствует финансирование, в т.ч. для обеспечения специалистов необходимым оборудованием и для их непосредственного участия в ежегодных встречах).

В СССР были созданы рабочие группы по стандартизации языков программирования, в т.ч. группа по стандартизации Фортрана. В Российской Федерации такая группа не сформирована, и работа одного эксперта по личной инициативе и только по электронной почте не позволяет более активно участвовать в международной деятельности.

По нашему мнению, было бы весьма полезным более тесная интеграция российских ученых в международную деятельность по развитию и стандартизации

Фортрана – языка, который лидирует при решении больших вычислительных задач на современных высокопроизводительных ЭВМ.

В.2. По второму кругу проблем.

Исследования по данному направлению, финансируемые РФФИ, проводятся в нашей стране с учетом исследований, проводимых за рубежом. В то же время нельзя говорить о существенном влиянии российских специалистов на решение проблем в международном масштабе. Одна из причин заключается в том, что в нашей стране недооценивается роль унификации и стандартизации в международном масштабе разрабатываемых средств.

Для решения данного круга проблем, как и в предыдущем случае, целесообразна более тесная интеграция российских специалистов в международную деятельность. Это было бы полезно и для престижа страны, и для того чтобы в международных стандартах учитывались традиции, опыт отечественных специалистов и особенности задач, решаемых в нашей стране. Что касается прикладных задач вычислительного характера, решаемых на современных компьютерах, они должны быть написаны на стандартных языках с использованием современных технологий, которые обеспечивают эти стандарты. Это позволит создавать конкурентноспособный программный продукт (см. А, раздел 13).

Сентябрь 2002 года