СИСТЕМА ПРОГРАММИРОВАНИЯ ОТЕЧЕСТВЕННОЙ СЕРИИ СИГНАЛЬНЫХ КОНТРОЛЛЕРОВ «МУЛЬТИКОР»

(ОКОНЧАНИЕ. НАЧАЛО СМ. В ЭК №12, 2005)

Дмитрий Бочарников, технический директор, ЗАО «Интерстрон»
Игорь Замятин, ведущий программист, ЗАО «Интерстрон»
Станислав Крысенков, программист-разработчик, ЗАО «Интерстрон
Вадим Синицын, ведущий инженер-программист, «Элвис»

Во второй части статьи описаны инструментальные средства — ассемблер, линкер, библиотекарь, а также отладчик и интегрированная среда разработки для первой линейки отечественных DSP-контроллеров семейства «Мультикор», разработанной в ГУП НПЦ «Элвис» (г. Зеленоград).

АССЕМБЛЕР, ЛИНКЕР, БИБЛИОТЕКАРЬ

Макроассемблер для процессоров «Мультикор» предназначен для ассемблирования программного кода и данных для DSP- и RISC-ядер. Исходный текст на макроассемблере транслируется в два этапа. На первом работает макропроцессор, который имеет развитые макросредства для эффективного управления трансляцией программ. Эти средства позволяют программисту определять в тексте программы многострочные и однострочные макросы, числовые переменные, использовать условные директивы, циклы, макрокоманды для работы со строками и другие возможности, повышающие удобство разработки программ на ассемблере. На втором этапе производится трансляция преобразованного макропроцессором текста. Нужно отметить, что для каждого ядра процессора имеется собственная система инструкций.

При создании ассемблера и линкера преследовалась цель — минимизировать количество информации, которое нужно знать пользователю, чтобы писать программы с взаимодействием RISC- и DSP-ядер. Различие секций RISC и DSP на синтаксическом уровне заключается в наличии специального атрибута в заголовке секции, который позволяет ассемблеру понять, что он имеет дело с секцией определенного типа. Так, объявление .section ".text" ro означает, что объявлена RISC-секция кода, а .section ".text" ro d — DSP-секция кода.

В результате работы ассемблера создается перемещаемый объектный файл, содержащий секции обоих типов и обращения к символам, которые надо разрешить. Задача линкера заключается в том, чтобы разместить нужным образом все секции в адресном пространстве и разрешить обращения к символам. Линкер помещает DSP-секции кода и данных в специально отведенную для этого область памяти. При разрешении перемещаемых выражений линкер учитывает, в какой секции определен тот или иной символ и в какой секции встречается обращение к нему. Каждому символу, определенному в DSP-памяти, соответствует два адресных значения: абсолютный виртуальный адрес RISC-памяти и внутренний относительный адрес DSP-памяти. Дело в том, что DSP-ядро имеет собственную внутреннюю адресацию и содержит два независимых пространства — память данных и память программ. Если линкер встречает обращение в RISC-секции к символу, определенному в DSP-памяти, то в качестве значения символа берется его RISC-адрес, если же обращение к этому символу происходит из DSP-секции, то берется его относительное значение.

Линкер может обрабатывать несколько модулей, содержащих различные секции с одинаковыми именами. По умолчанию линкер размещает такие секции одну за другой в едином виртуальном адресном пространстве. При этом DSP- и RISC-секции размещаются каждая в своей области памяти. Для конкретизации размещения секций в адресном пространстве, а также для задания адресов общего адресного пространства для размещения RISC- и DSP-секций служит конфигурационный файл линкера.

Помимо размещения секций, конфигурационный файл поддерживает механизм оверлеев, который позволяет преодолеть ограниченность и небольшой размер памяти DSP-ядра по сравнению с RISC-памятью. Распространенная ситуация для многих классов задач заключается в том, что размер памяти, необходимой для кода и данных, превышает объем DSP-памяти. Каждая секция в программе может загружаться с одного адреса, а выполняться с другого. Линкер отслеживает такую двойственную логику распределения секций на этапах загрузки и исполнения и при разрешении перемещаемых выражений вместо символов подставляет адреса по исполнению. В конфигурационном файле для выделенных областей памяти, в которых будут размещаться определенные секции, можно приписать явные адреса размещения и исполнения. При анализе конфигурационного файла линкером будут сгенерированы специальные метки начала и конца таких областей по адресам размещения и исполнения. При помощи этих меток можно по мере надобности подгружать одни DSP-секции из RISC-памяти в DSP и отгружать другие из DSP в RISC, тем самым эффективно используя ограниченную память DSP-ядра.

В конфигурационном файле можно задать описания максимум трех разделов адресного пространства (RISC, DSP_PRAM, DSP_XYRAM), а в каждом из разделов — определить ряд областей, в которых будут располагаться секции. Каждая область характеризуется своим именем, адресом размещения, адресом исполнения и максимально возможным размером (два последних атрибута задавать необязательно). В описание области также входит набор признаков, по которым ту или иную секцию можно отнести именно к данной области определенного раздела. Секцию можно отнести в какую-то область либо по имени, либо по месторасположению.

Для разделов RISC и DSP_XYRAM можно также задать размеры стека и кучи.

Ниже приведен пример конфигурационного файла:

```
RISC
stacksize 0x10000
heapsize 0x10000
area “Area1” Ar = 0xbfc00000 MaxSize = 0x500
“.text1”
“.data1” (“f1.obj” “lib1.lib”:”f2”,”f3” “f4.obj”)

area “Area2” Ar = 0xbfc00500
default

DSP_PRAM
area “Area3” Ar = 0xbfd00000 Ai = 0xb8440000
“.textdsp1”

area “Area4” Ar = 0xbfd05000 Ai = 0xb8440000
“.textdsp2”

area “Area5” Ar = 0xb8441000
default
```

Здесь в RISC-разделе выделены две области — Area1 и Area2. В первую область попадут все секции с именем .text1, а также секции .data1 из модулей f1.obj, f4.obj и библиотечных модулей f2, f3. Во второй области размещаются все остальные RISC-секции. В разделе PRAM определено 3 области, при этом для Area3 и Area4 задан адрес размещения и исполнения. Эти области являются оверлейными, так как при исполнении секции каждой области (.textdsp1 и .textdsp2) будут располагаться в памяти последовательно друг за другом с одного и того же адреса 0xb8440000. Однако при начальной загрузке исполняемого файла и в те моменты, когда от секций не требуется нахождение в DSP-памяти, секции каждой области размещаются в непересекающихся участках памяти. Сгенерированные линкером символы для обозначения начала и конца области Area3 по размещению и исполнению будут следующими:

```
___areaDsp_r_Area3__base,
___areaDsp_r_Area3__limit,
___areaDsp_i_Area3__base,
___areaDsp_i_Area3__limit.
```

Линкер дает возможность анализа структуры сформированного исполняемого файла, в том числе просмотра состава секций и символов. Для этой цели при вызове линкера необходимо задать опцию, по которой будет создан специальный map-файл, содержащий информацию обо всех секциях и нелокальных символах исполняемого файла. Для каждой секции указываются ее тип, атрибуты, адрес размещения в памяти, размер и имя объектного модуля, из которого она была взята. Для символа указываются его тип и адрес.

Кроме основного режима работы — создания исполняемого файла — линкер также может работать и в режиме библиотекаря, то есть создавать и модифицировать библиотеки объектных модулей. В библиотеку могут входить как RISC-, так и DSP-секции.

ОТЛАДЧИК

Хорошо известно, что в жизненном цикле программной системы наибольшую трудоемкость составляет процесс отладки. Поэтому все современные системы разработки обязательно содержат развитые средства отладки, призванные упростить и облегчить этот процесс. В настоящее время мощный отладчик в терминах исходной программы является стандартной компонентой развитых средств разработки на языках высокого уровня. Типичный «репертуар» средств отладки включает установку условных или безусловных точек останова, пошаговую трассировку с привязкой к исходному тексту, просмотр и изменение значений объектов программы.

Отладчик компании «Интерстрон» для платформы «МУЛЬТИКОР» предназначен для загрузки и отладки программ, полученных при помощи инструментальных средств (компилятора, ассемблера, линкера). Отладчик реализован в виде законченной самостоятельной программы, которая может работать как отдельная консольная утилита. Кроме того, отладчик интегрирован в среду разработки (см. след. раздел).

Пользовательским интерфейсом отладчика служит известный интерфейс GDB/MI, разработанный сообществом GNU (www.gnu.org), и к настоящему времени ставший стандартом де-факто. Привлекательной особенностью отладчика служит его сравнительно слабая зависимость от операционной системы инструментального компьютера, поэтому его портирование на другую ОС не представляет большой проблемы.

Отладчик выполнен в виде независимого ядра, отвечающего за загрузку кода и исполнение команд пользователя, и специализированных библиотек, реализующих эмуляцию целевого процессора. Подключение библиотек производится через специальный программный адаптер между интерфейсом библиотеки и интерфейсом отладчика. Такая архитектура обеспечивает большую гибкость отладчика и сравнительно несложную настройку на различные процессоры. При работе отладчика библиотеки эмуляции могут загружаться как статически, так и динамически.

Формат отладочной информации

В качестве формата отладочной информации был выбран DWARF версии 3 (DWARF Working Group). На сегодняшний день DWARF является одной из лучших спецификаций отладочной информации для языков программирования высокого уровня. Он предоставляет возможность полностью описать все сущности языка C++ — переменные, функции, классы, сложные структуры, указатели их областей видимости, а также задать их привязку к целевому коду. Дополнительным аргументом в пользу выбора формата DWARF служит его положение как стандарта де-факто в своей области, что дает возможность при необходимости использовать отладочные средства сторонних производителей.

Генерация информации для отладчика реализована таким образом, что для отладки нет необходимости создавать особую «отладочную версию» программы. Генерация отладочной информации не влияет на размер конечного кода, так как отладочная информация помещается в промежуточные носители (в файл с ассемблерным кодом, в файл с объектным кодом, а также в специальный файл, содержащий собственно отладочную информацию). Отладка происходит непосредственно на том коде, который будет загружаться в целевой процессор.

Отладка оптимизированного кода

Поддержка соответствия исходному тексту при отладке оптимизированного кода составляет одно из важнейших преимуществ данной реализации. Была проделана большая работа, чтобы наиболее полным образом соотнести исходный текст и сгенерированный оптимизированный код.

Так, отладочная информация позволяет отследить ситуацию с «поднятием» переменных в регистры, что показывает следующий пример:

```
int a[5] = {1, 2, 3, 4, 5};

int f()
```

```
{
    int i, sum;
    sum = 0;
    for (i = 0; i < 5; ++i) {
    sum += a[i];
    }
    return sum;
}

int main()
{
    return f();
}
```

В процессе выполнения цикла значения переменных i и sum находятся в регистрах, а не в памяти. Это можно проверить по ассемблерному тексту (см. след. подраздел). Тем не менее, в процессе отладки в интегрированной среде значения переменных показываются правильно, так как отладчик сам «знает», откуда их брать.

К сожалению, обеспечение полного соответствия для оптимизированного кода возможно не во всех случаях. Например, не всегда удается отразить результаты таких оптимизаций, как пронос констант или удаление «мертвого кода», что иллюстрирует следующий пример:

```
int f( int i )
{
    const int a=1;
    const int b=2;
    int c;

    c = a + b;
    return i+c;
}
```

Для этой функции фактически будет скомпилирована одна команда: **return** i+3. Так как переменная с реально не используется, то команды ее вычисления в объектном коде нет. Поэтому в процессе пошагового выполнения в отладчике при входе в данную функцию указатель выполняемой команды сразу будет установлен на строке **return** i+c, а переменную с нельзя будет посмотреть. Возможность подобных ситуаций следует учитывать при работе с оптимизированным кодом.

Отладка на уровне ассемблера

Как и большинство современных систем программирования, пакет средств разработки компании «Интерстрон» для платформы «МУЛЬТИКОР» позволяет вести отладку на уровне ассемблера, включая установку точек останова, пошаговую трассировку с привязкой к ассемблерному тексту, просмотр и изменение регистров и памяти. Отладочная информация для нее создается ассемблером. Исходный текст представлен в виде комментариев, что позволяет ориентироваться в тексте программы. При описании отладки оптимизированного кода мы уже обращались к отладке на уровне ассемблера, чтобы понять, как реализована программа. К примеру, упомянутый выше фрагмент программы выглядит так (неоптимизированный код):

```
. . .
;   sum = 0;

    addiu $3, $0, 0
    sw $3, -4($30)

; for(i = 0; i < 5; ++i) {

L2:
    addiu $4, $0, 5
    lw $3, -4($30)
    nop
    sub $3, $3, $4
    bltz $3, L3
    nop
    j L4
    nop

; sum += a[i];

L3:
    lw $4, -4($30)
    nop
    la $3, _a
    addiu $5, $0, 4
    mul $4, $4, $5

. . .

    j L2
    nop

; }
; return sum;

L4:

. . .
```

Таким образом, разработчику предоставляется возможность производить отладку на любом уровне. Неоптимизированный код, полностью соответствующий исходному, можно использовать для отладки логики самой программы, не обращая внимания на особенности реализации. Оптимизированный код позволяет протестировать конечный результат. Отладка на уровне ассемблера позволяет обратить внимание на тонкости реализации программы для целевого процессора. Данные возможности позволяют разработчику значительно сократить время отладки и повысить безопасность разрабатываемого программного обеспечения.

Интегрированная среда

Использование компилятора, ассемблера, отладчика и других средств в виде независимых утилит с их вызовом из командной строки существенно снижает производительность труда разработчиков и качество результирующих программ. Следовательно, одна из наиболее существенных и первоочередных задач при разработке инструментального ПО — создание развитой среды разработки, интегрирующей все ключевые компоненты для их совместной работы и удобства разработки.

В качестве основы такой интегрированной среды была выбрана универсальная компонентная платформа Eclipse (www.eclipse.org). Сейчас платформа Eclipse активно развивается; в сообщество разработчиков Eclipse входят такие компании, как Intel, Borland, QNX и другие.

Пользовательская среда программирования для процессоров «Мультикор» является одним из примеров использования этой технологии компанией «Интерстрон».

Интегрированная среда разработки поддерживает следующие основные виды работ.

— **Разработка программных проектов на языках C/C++ и языке Ассемблера**. Поддерживается полнофункциональный текстовый редактор программ, включающий стандартный набор средств, в частности, синтаксическую подсветку (в том числе и для ассемблерного кода). Реализован менеджер проектов, обеспечивающий процесс автоматической сборки проектов с позиционированием ошибок и предупреждений по исходным текстам программ.

— **Компиляция программ и проектов на языках C/C++ и Ассемблере**. Менеджер проектов позволяет создавать различные конфигурации построения целевых программ, а также адекватно обрабатывать параметры компиляции, ассемблирования и линковки, в том числе опции, предназначенные специально для данного микропроцессора.

— **Отладка в графической среде с помощью отладчика**. Поддерживается навигация по исходному коду в процессе отладки, просмотр и изменение значений ресурсов целевого процессора — памяти, переменных, регистров. Имеется возможность вычисления значения выражений, заданных

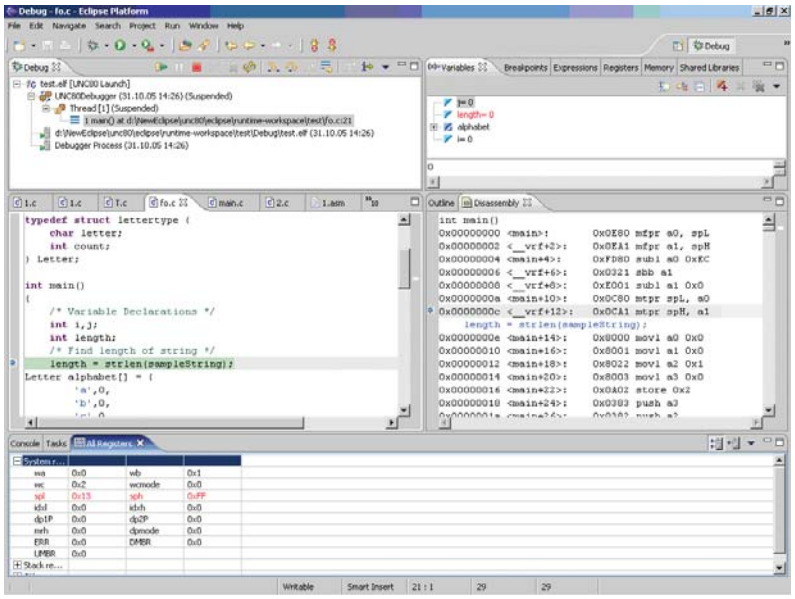

Рис. 1. **Окно интегрированной среды разработки и отладки программ**

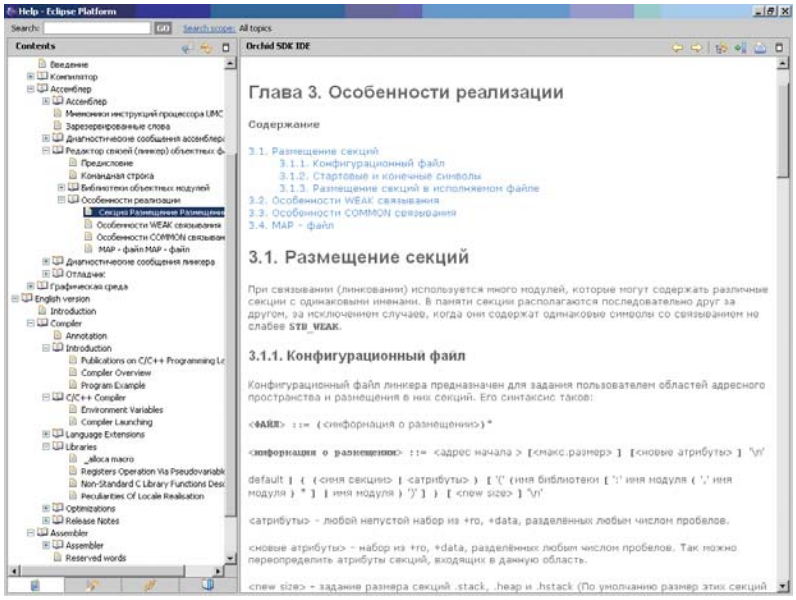

Рис. 2. **Окно подсистемы оперативной помощи**

в синтаксисе языков C/C++. Также реализованы установка точек останова, навигация по ассемблерному тексту, отображение стека вызовов и навигация по нему.

Учитывая сказанное, следует отметить, что на основе собственной технологии интеграции отладчика со средой разработки компании «Интерстрон» удалось создать полнофункциональный инструмент для разработки и отладки программ на языках C/C++ для платформы «МУЛЬТИКОР» (см. рис. 1). Эта технология минимизирует время разработки среды, а также затраты, связанные с ее переносом на другие ОС.

– **Подсистема оперативной помощи**. Высокая степень структурированности материала (а также его двуязычности) в сочетании со стандартными для оперативной помощи средствами навигации и поиска обеспечивают удобную и эффективную поддержку для разработчиков (см. рис. 2).

Все инструментальное ПО заключается в инсталляционный пакет, при помощи которого пользователь может легко установить программное обеспечение и начать работу без каких-либо предварительных настроек.

ЗАКЛЮЧЕНИЕ

Завершая описание инновационного инструментального ПО для платформы «МУЛЬТИКОР», следует отметить, что результатом совместной работы ЗАО «Интерстрон» и ГУП НПЦ «ЭЛВИС» стал пакет инструментальных средств для разработки и отладки программ для процессоров серии «Мультикор», названный «MCStudio-ECL».

Центральной частью инструментальных средств служит компилятор полного стандарта для языка C/C++ фирмы ««Интерстрон»». Ключевой особенностью комплекса является реализация в рамках единого компилятора механизма создания выполняемых файлов кода для различных ядер, входящих в состав MIPS32-совместимой архитектуры процессоров «Мультикор». В данной версии системы процесс загрузки ядер процессора, соответствующий той или иной микросхеме серии «Мультикор» осуществляется программистом-разработчиком на этапе создания программного проекта.

Пакет средств разработки, поставки которого пользователям микросхем «Мультикор» начнутся уже в первом полугодии 2006 г., содержит:

– пользовательскую среду разработки программ на C/C++, реализованную для режимов создания и отладки;

– компилятор программ на C/C++, реализующий стандарты ISO/IEC 9899:1999(C99) языка C и ISO/IEC 14882:1998 языка C++;

– объединенный Ассемблер для RISC- и DSP-ядер процессоров серии «Мультикор»;

– линкер выполняемого файла и библиотекаря для создания пользовательских библиотек;

– системные библиотеки языков C и C++;

– отладчик программ C/C++.

Продукт «MCStudio-ECL» оперирует со всеми разработанными ранее и поставленными пользователям отладочными комплектами для микросхем 1892ВМ2Т и 1892ВМ3Т. Это позволяет выполнить по желанию пользователей дооснащение их инструментальных средств новым продуктом.

Кроме того, выход на рынок такого продукта, как «MCStudio-ECL» дает мощный импульс замещению импортной элементной базы отечественной, т.к. решает проблему переноса ранее наработанного ПО с ряда зарубежных DSP-платформ (таких как TMS 320xx фирмы Texas Instruments или Tiger SHARC фирмы Analog Devices) на отечественную платформу «МУЛЬТИКОР».

В работе над проектом принимали активное участие: Е.А. Зуев, М.Ю. Донорский, А.А. Чупринов, Ю.Н. Александров, В.Ф. Никольский, Т.В. Солохина, Я.Я. Петричкович, Беляев А.А., Глушков А.В.