Управление компактором и оптимизатором G21k v3.3.

Иногда необходимо добавить дополнительный уровень управления оптимизатором и компактором, так как оба могут изменить код, что бывает в приложении не приемлемым. Это может быть вызвано и необходимостью уменьшения кода, чтобы он помещался во внутреннюю память DSP. Этот документ содержит некоторые предложения и методы, которые позволяют настроить функции компактора и оптимизатора под требования пользователя.

Вначале прочитайте замечания по средствам разработки для ADSP-21000 PC версии 3.3 и UNIX версии 3.3.1 («ADSP-21000 Family Development Tools Release Note for 3.3 PC Release and 3.3.1 UNIX Release»). Этот документ содержит раздел

Что делает оптимизатор?
Оптимизатор оптимизирует только исходный код на языке С.

Что делает компактор?
Компактор оптимизирует код ассемблера. Это может быть код, созданный С компилятором или существующий исходный код на языке ассемблера. Компактор имеет свой набор приемов, чтобы уменьшить размер кода и сделать его быстрее.

Сценарий 1:

Я использую только оптимизатор, и он меняет мой С код так, что он не работает корректно.

Нет возможности выборочно отключить оптимизатор, если у вас возникли проблемы. Ниже приводятся некоторые приемы, чтобы заставить оптимизатор вести себя правильно.

1) Проверьте, что ваш код работает правильно без использования оптимизатора.

2) Если ваш код слишком большой, чтобы поместиться в отведенной памяти без оптимизатора, создайте другой файл архитектуры для системы с большей памятью. Откомпилируйте вашу программу с новым файлом архитектуры и запустите его в программе отладки.

3) Если вы уверены, что это точно оптимизатор, следуйте следующими шагами.

4) Объявите все свои статические переменные (*static variables*) как непостоянно статические (*volatile static*). Одно из того, что делает оптимизатор и может вызвать большинство проблем, это то, что он оптимизирует вне переменных. Команда *volatile* гарантирует, что они будут защищены.

5) Если вы используете сложные логические условия в командах *if, for, while*, попробуйте переставить их. Иногда оптимизатор неправильно обрабатывает сложные логические условия.

6) Используйте ключ *–mlistm* вместе с компилятором. Это создаст файл, который будет содержать каждую С инструкцию с ассоциированной инструкцией на ассемблере. Здесь вы сможете проверить правильность компилированного кода. Возможно, это покажется слишком долгим - проверять тысячи строк кода. Старайтесь проверять сложные конструкции.

7) Если у вас нет идей, где в исходном коде находится источник проблемы, одним из выходов может послужить расстановка в исходном тексте «ловушек». Так как вы не можете пользоваться CBUG при использовании оптимизатора, старайтесь изолировать проблему. Добавляйте и удаляйте «ловушки» в вашем коде, так чтобы видеть, как долго работает код перед сбоем.

Сценарий 2:

Я использую оптимизатор и компактор одновременно, и мой С код не работает корректно.

1) Во-первых, проверьте, что ваш код работает и без компактора с оптимизатором.

2) Проверьте, что ваш код работает только с оптимизатором, если нет, см. первый сценарий.

3) Откомпилируйте ваш код с ключом *–save-temps* и ключом *–mlistm*. Ключ *–save-temps* сохранит промежуточные файлы с расширением «.s», «.i» и «.is». Файлы с расширением «.s» содержат код на языке ассемблера, созданный компилятором. Ключ *–mlistm* вставляет в этот файл ваш исходный С код как комментарий.

4) Если компилятор выдает сообщение об ошибке, что возникла ошибка в «.is» или «.s» файле, перейдите к указанной строке, в которой должен быть комментарий из вашего кода на языке С. Перейдите к пункту 6.

5) Если комментарий не выдает сообщение об ошибке, необходимо изолировать место, которое не ассемблируется корректно. Используйте методы из сценария 1.

6) Когда вы изолировали код, который компилируется некорректно, отключите работу компактора в этих строках.

В вашем исходном коде используйте следующие функции для разрешения и запрещения работы компактора.

Для разрешения: **asm(«!.start_dco»);**
Для запрещения: **asm(«!.end_dco»);**

Вставьте эти строки вокруг проблемного кода:

```
asm(«!.start_dco»);
// правильно компилируемый код
int a;
float b,c;
...

asm(«!.end_dco»);
// неправильно компилируемый код
for (I=0; I<10; I++)
      {
            a+=10*c;
      }
...
asm(«!.start_dco»);
// правильно компилируемый код
b = b * c;
...
asm(«!.end_dco»);
```

Теперь код готов к компиляции. Необходимо вызвать компилятор дважды, чтобы это работало. При первом вызове создается код на языке ассемблера. Не забудьте про указание ключей *–s,–save-temps* и *–mlistm*. Второй вызов компилирует код на языке ассемблер, разрешая компактор.

```
g21k cmain.c -a 21062c.ach -nomem
   -S -save-temps -mlistm
g21k cmain.i -Wo,-slct -O3
   -a21062c.ach -o cmain.exe
```