**Руководство пользователя системы разработки программ MicroGenSf для микропроцессоров**

В комплект программ входят:

|  |  |  |
| --- | --- | --- |
| **TCP.EXE** | - | оптимизирующий компилятор с языка "С" V 5.20, |
| **TCLN.EXE** | - | редактор связей ("связывающий загрузчик"), |
| **TCLIB.EXE** | - | библиотекарь. |
| **CLIB.REL** | - | библиотека стандартных функций |
| **\*.H** | - | стандартные заголовочные файлы. |
| **\*.C, \*.MAC** | - | исходные файлы библиотеки |
| **\*.BAT** | - | командные файлы для создания библиотеки. |

Файлы перемещаемого кода, используемые перечисленными программами, имеют расширение .REL. Знакомым с операционными системами CP/M-80, ОС-1800, МИКРОДОС, MSX-DOS это напомнит недавнее прошлое, а многих - обрадует. Дело в том, что это действительно файлы перемещаемого кода в формате, предложенном фирмой Microsoft, и используемом всеми ее компиляторами для восьмиразрядных машин. По этой причине можно использовать при разработке прикладных программ известные в системе CP/M программы M80.COM, L80.COM, LIB80.COM, а так же RMAC.COM и LINK.COM разработки фирмы Digital Research.

Используемый формат определяет ограничения на внешние имена, используемые Вашими программами: редактору связей известны первые 7 символов имени, при этом имеет значение, большими или маленькими буквами написано имя.

# 1. Ограничения данной версии:

Компилятор версии 5.20, имеет следующие ограничения:

Несмотря на то что при описании типов допустимы все обозначения (char...float), арифметические операции и инициализация в описаниях допустимы только над char, int, long (только присваивание, передача в качестве параметра, возврат из функции, +, -, &&, ||, &, |, ~, ^) и адресами. Для long дополнительно допустима инициализация в описаниях, при этом инициализатор должен иметь допустимое значение для signed или unsigned int.

Передача по значению struct и union не поддерживается. Так же, как и присваивание с использованием этих типов.

Не поддерживается продолжение макроопределений и строковых констант на следующей строке с помощью символа обратной косой черты. Для получения возможности продолжения строковой константы на следующей строке нужно пользоваться записью типа: "строка1" "строка2", что эквивалентно записи типа: "строка1строка2".

**Внимание!** Не путайте строковые константы с очень похожими по записи инициализаторами символьных массивов – для них нельзя использовать предложенный способ продолжения.

Не поддерживаются макроопределения с параметрами.

Не поддерживается инициализация в описаниях автоматических объектов.

# 2. Опции компилятора

Общий формат командной строки компилятора:

**TCP [возможные опции] файлы**

Опции бывают с параметрами и без них. Несколько символов опций могут стоять в строке рядом без пробелов, но опция, требующая задания параметра, должна завершаться пробелом. Каждой опции или группе опций должен предшествовать символ '-' (минус). Файлы, имена которых не должны начинаться со знака 'минус', следуют друг за другом через любое количество пробелов. Расширением исходных файлов по умолчанию становится '.C', если явно не задано другое. Выводные файлы перемещаемого кода (для каждого исходного) всегда получают расширение '.REL'. Перечисление нескольких файлов в командной строке - просто компиляция последовательно каждого из них, без перезагрузки компилятора.

Автоматический вызов редактора связей не предусмотрен, пользователь должен выполнять его вызов явно (в ручную или из командного файла).

В случае обнаружения ошибок во время сеанса компиляции группы файлов (далее - компиляции) компилятор при своем завершении устанавливает отличное от 0 значение ERRORLEVEL, что позволяет прекратить компиляцию после ошибок. Это можно использовать в .ВАТ файлах или через МАКЕ.

Итак, ключи компилятора:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **-dname** | Эквивалентно строке #define name 1 | | | |
| **-q** | Использование вызовов подпрограмм вместо встраивания в программу последовательности команд для выполнения арифметических операций над 16-битными значениями. Рекомендуется для получения более компактного кода. | | | |
| **-z** | При генерации программы используются команды микропроцессора Z80/U880. Используются команды косвенной пересылки байтов по смещению от IX, 16-битное вычитание. Второй набор регистров, регистр IY, короткие переходы, битовые проверки и остальные "прелести" не используются.  За счет прямого встраивания команд в код (чтение/запись 16-битного числа из автоматических переменных занимает 6 байт) и появляющихся в начале каждой процедуры команд сохранения содержимого регистра IX и загрузки в него нового значения ('базовый адрес') объем скомпилированной программы увеличивается. Но скорость работы существенно возрастает. | | | |
| **-k** | Тип char знаковый, в противном случае описание char эквивалентно unsigned char. | | | |
| **-m** | Ключ, используемый при компиляции библиотек. При его использовании после завершения компиляции очередной функции с классом 'extern' завершается формирование очередного модуля в объектном коде, классы всех ранее определенных объектов заменяются на 'extern' и с нового объекта начинается генерация очередного модуля. Об особенностях генерации и использования библиотек - в разделе описания программы TCLIB.EXE. | | | |
| **-O** | Ключ для запрещения оптимизации - в основном используется при тестах на скорость и отладке. При обычной работе не рекомендуется его применять, так как объем генерируемого кода увеличивается почти в два раза. | | | |
| **-E#** | На месте символа '#' должно стоять число - оно определяет максимальное количество поименованных объектов (переменных, функций, типов, констант, аргументов у функций), используемых в программе.  Некоторые количества используемых объектов: | | | |
|  |  | **int test(int a, int b);** | - | 3 имени |
|  |  | **int cmp(char \*, char \*);** | - | 3 имени |
|  |  | **enum colors { RED, GREEN, WHITE, BLACK } TCL;** | - | 6 имен |
| **-l#** | Эту опцию придется применять, если общий объем строковых констант в одной функции превышает 400 байт. Текст этих констант помещается за функцией, но до закрывающей скобки он хранится в буфере. | | | |
| **-g#** | После указанного '#' количества предупреждений компиляция прекращается. | | | |
| **-j#** | После указанного '#' количества ошибок компиляция прекращается. | | | |

# 4. Распределение памяти.

Скомпилированная программа состоит из двух сегментов: данные и код.

В сегмент кода, модификация содержимого которого в случае размещения готовой программы в ПЗУ невозможна, помещаются все исполняемые части программы (тела всех функций), строковые константы, инициализированные объекты (переменные, массивы, и т.д.).

В сегмент данных, располагаемый в области ОЗУ, попадают все неинициализированные объекты.

Соглашение об умалчиваемой инициализации объектов, не имеющих явной инициализации, для встраиваемых систем автоматически не выполняется. Решается эта проблема (при необходимости) заполнением кодом 0х00 всей области ОЗУ из стартового модуля. При получении обычной .СОМ программы для исполнения под СР/М все неинициализированные объекты заполняются байтом 0х00 на этапе редактирования связей.

# 5. Подготовка стартового модуля для целевой компиляции.

Поставляемая библиотека CLIB.REL содержит в своем составе стартовый модуль (по имени CROOT), выполняющий начальные установки для выполнения программы под управлением СР/М. При необходимости получения программы, работающей в другом окружении, рекомендуется придерживаться следующих рекомендаций к аппаратным и программным средствам:

* ПЗУ располагайте с адреса 0х0000.
* После сброса управление сразу передается на метку '?START'.
* ОЗУ может располагаться в любом месте, но желательно 'без дыр'.
* Программа не должна завершаться.
* Не инициализировать периферийные микросхемы в стартовом модуле.
* Ассемблерные функции использовать только там, где иначе нельзя.

Пример текста стартового модуля для системы со следующим распределением памяти:

0x0000..0x1fff ПЗУ объемом 8 КБ.

0x3800..0x47ff ОЗУ объемом 4 КБ.

**;**

**; crom.mac**

**;**

**public $memry, EDATA\_, BDATA\_**

**public ?start**

**extrn rroot**

**;**

**?start: lxi sp,4800h**

**lxi h,0**

**BDATA\_ equ $ - 2**

**$memry equ $ - 2**

**shld EDATA\_**

**call rroot**

**jmp ?start ; Loop forewer.**

**;**

**dseg**

**EDATA\_: dw 0**

**cseg**

**;**

**end ?start**

**Комментарии:**

Главная функция Вашей программы должна быть описана так:

**void pascal rroot(void);**

Атрибут 'pascal' обозначает, в данном случае, что внешнее имя этого объекта передается всегда буквами верхнего регистра и усеченным до 6 символов - т.е. приведенным к соглашениям о внешних именах ассемблера M80.COM фирмы Microsoft. Остальные особенности ниже, в разделе о передаче параметров функциям.

Именем '$MEMRY' помечены 2 байта в коде, в которые после связывания целевой программы помещается адрес первого свободного байта в сегменте данных. Копирование этого значения в другие переменные нужно для правильной работы функций распределения памяти brk() и sbrk(). Так же к используемым функцией brk() относятся переменные 'BDATA\_' и 'EDATA\_'. Соглашение о имени '$MEMRY' поддерживается TCLN.EXE и L80.COM.

Часть стартового модуля, заполняющая все ОЗУ кодом 0x00, нужна для выполнения упомянутого выше соглашения об инициализации переменных.

# 5. Использование ассемблера.

При написании ассемблерных функций необходимо знать:

Переменные и функции в Вашей "С" программе, если хотите использовать их в ассемблерной функции, должны иметь атрибут 'pascal' либо иметь имя длинной до 5 букв и следом подчеркивание (NAME\_) при этом в С – модуле они должны быть описаны большими буквами без подчеркивания, при это параметры передаются в С стиле

**Пример:**

**int pascal vara, varb;**

**void pascal pfunc(void);**

**int \* pascal pfunc(int);**

**int (\* ppfp)(char \*);**

**int PFUN2(char \*);**

**Комментарии:**

|  |  |  |
| --- | --- | --- |
| **vara** | - | pascal переменная |
| **varb** | - | cdecl переменная |
| **pfunc** | - | pascal функция |
| **ppfp** | - | pascal указатель на cdecl функцию |
| **PFUN2** | - | С функция |

Для функций, не имеющих атрибута 'pascal', аргументы вычисляются и помещаются в стек в порядке "справа налево".

**Пример:**

Следующий текст транслируется так:

**void pascal test(int, int);**

**test(0, 100);**

**lxi h,0**

**push h**

**lxi h,100**

**push h**

**call test**

**pop d**

**pop d**

Следующий текст транслируется так:

**void test1(int, int);**

**test(0, 100);**

**lxi h,100**

**push h**

**lxi h,0**

**push h**

**call test1**

**pop d**

**pop d**

Всеми функциями значение результата (если необходимо) возвращается в вызывавшую программу в регистровой паре HL, но если функция описана как возвращающая значение типа char или unsigned char, то значение возвращается в регистре A .

При возврате функция должна восстановить:

* Содержимое регистровой пары BC.
* Содержимое регистра IX (если используется процессор Z80/U880).
* Значение указателя стека SP.

Регистровая пара BC используется для хранения переменных 'register'. Содержимое регистра IX используется для доступа к автоматическим переменным.

В качестве примера написания ассемблерных функций, вызываемых из "С" - программ, может служить пакет работы с памятью - файлы MEM.MAC и MEM.H.

В систему разработки не включен компилятор с языка ассемблера, для компиляции можно пользоваться M80.COM, запускаемым под эмулятором операционной системы СР/М.

# 6. Программа TCLIB.EXE

В системе поставляется программа для работы с объектными модулями. Для пояснения некоторых ее функций требуется небольшое отступление:

Так как компилятор реализован по однопроходной схеме, полный список имен для каждого модуля становится известен к концу чтения исходного файла и, соответственно, к концу формирования выходного. Но редакторам связей (TCLN.EXE, L80.COM) эта информация необходима в начале модуля. Операцию переноса списка имен в начало модуля выполняет библиотекарь TCLIB.EXE для каждого модуля, с которым он имеет дело. Ассемблер формирует все как надо, т.к. осуществляет трансляцию за 2 прохода. Но это преобразование требуется только для модулей (библиотек), в которых осуществляется поиск необходимых глобальных имен. Для 'связывания' годятся модули непосредственно после компиляции. Информация: компилятор с языка 'BASIC' фирмы Microsoft, работающий на СР/М-80, так же выполнен однопроходным.

Опции программы TCLIB.EXE:

|  |  |
| --- | --- |
| **-l** | Печать имен и размеров модулей, глобальных имен, определенных в каждом из обрабатываемых модулей. |
| **-q** | Дополнительно печать глобальных имен, Неопределенных в каждом из обрабатываемых модулей. |

За опциями следуют:

Одно имя - файл преобразуется, как описано выше.

Более одного имени - первое встретившееся имя обозначает выходной файл, в который копируются (с описанными преобразованиями) модули из остальных файлов.

Расширение по умолчанию для каждого исходного файла - .REL, если явно не задано другое. Старая версия исходного файла уничтожается только после успешного завершения всех операций по записи нового файла. Расширение имени нового файла по умолчанию будет .REL, если явно не указано другое.

# 7. Редактор связей TCLN.EXE

Редактор связей преобразует файлы перемещаемого формата .REL в двоичный образ программы, настроенный на конкретные адреса начала сегментов кода и данных. Результатом работы является двоичный файл, содержащий образ программы. Формируется еще один файл (с расширением .SYM и именем, совпадающим с выходным) - текстовый файл соответствий между определенными в программе символическими именами и абсолютными адресами пространства памяти той машины, на которой будет исполняться созданная программа.

**Опции:**

|  |  |
| --- | --- |
| **-lname** | Имя файла 'name', в котором производится поиск модулей, к точкам входа которых есть неудовлетворенные ссылки, оставшиеся после связывания перемещаемых файлов, перечисленных в командной строке программы. |
| **-onam**e | Имя файла 'name', в который будет записан результат работы связывающего загрузчика. Расширением имени этого файла по умолчанию становится .COM. Одновременно образуется файл со списком символических имен программы. Его расширением всегда является .SYM. |
| **-pxxxx** | Шестнадцатеричное число 'xxxx' указывает загрузчику, с какого адреса размещать сегмент исполняемых кодов программы. |
| **-dxxxx** | Шестнадцатеричное число 'xxxx' указывает загрузчику, с какого адреса размещать сегмент неинициализированных данных программы. Инициализированные данные попадают в сегмент кода программы. |

Если не заданы опции **-p** и **-d**, в начале получившегося образа программы резервируются 3 байта, в которые загрузчиком помещается машинная команда перехода к началу программы (метке '?START'). Этот режим используется для получения исполняемых программ для операционных систем СР/М и MSX-DOS.

# 99. Мелочи.

Программа для CP/M, занимающая минимальный объем:

**#include <dos.h>**

**void \_setargv(void) {}**

**void exit(int code) { \_exit(code); }**

**int**

**main(void)**

**{ bdos(9, (unsigned) "Hello, world!\r\n$");**

**return (0);**

**}**