**Программа CP/M EXEcutor для DOS v0.8**

**cpm.exe**

Copyright (C) 1989-2012 Keiji Murakami {К.Мураками}

**0. Введение**

cpm.exe - это эмулятор HD64180/Z80 + CP/M -80 2.2, работающий в MS-DOS.

В 1989 году был создан инструмент разработки CP/M для использования с PC-9801 без уже исчезнувшей платы эмулятора CP/M. 10 Гц 80286 в то время работал с достаточно приемлемой скоростью, но нынешний ПК может обеспечить намного более быструю среду CP/M, чем реальный Z80.

Он имеет следующие особенности.

1. MS-DOS общего назначения
2. Запуск программы CP/M непосредственно из командной строки DOS
3. Может входить и работать в среде CP/M с xccp
4. Также может быть использован режим эмуляции 8080 процессором V30
5. Гораздо быстрее, чем настоящая машина с современным ПК

Кроме того, следующие изменения были внесены после V0.7.

1) Поддержка HI-TECH C

* Исправлена ошибка неопределенной инструкции Z80.
* Продолжение выполнения нереализованного вызова BDOS.
* Если расширение .COM, объявить версию как 122h (MP/M)

2) Поддержка MBASIC

* Изменен список инструкций JP в разделе ввода BIOS.

2) Улучшен метод обнаружения V30

* Используется AAD 0
* 8086 больше не идентифицируется как V30 (так и должно быть).

3) Скорость настроена для процессора 486 или выше

* Устранены инструкции обмена между памятью и регистрами.
* Разобраны некоторые инструкции lods.
* Повышена скорость на 20-60%.

4) Меры для программ, которые всегда имеют доступ к диску A:

* Если диск A: съемный, переключите доступ на C:

5) Прочие

* Файлы с расширением ".COM" также являются исполняемыми.
* Исправление орфографии EXEcuter-> EXEcutor (стыдно)

**1. Рабочая среда**

MS-DOS V3.1 или выше или совместимая среда

Эмулятор работает с 8086/8, 80186, V30, но мы рекомендуем использовать 80286 или выше, если это возможно. В случае V30 можно использовать режим эмуляции 8080.

Работа протестирована проверена на

* MS-DOS на ПК-9801VX
* DR-DOS на телефонном терминале NX91
* PC-DOS/V на AT -совместимом компьютере
* OS/2 в окне DOS на AT-совместимом компьютере
* Windows в окне DOS на AT -совместимом компьютере
* Windows 95, 98 в окне DOS на AT-совместимом компьютере
* Windows NT, 2k, XP в командной строке на AT-совместимом компьютере
* Linux в среде DOS Emu на AT-совместимом компьютере

**2. Использование**

Сначала перенесите программу CP/M в MS-DOS. В настоящее время, если задано расширение ".cpm" вместо".com", оно не будет ошибочно принято за исполняемый файл DOS.

Для запуска программы CP/M-80 первым аргументом cpm.exe укажите **имя программы** CP/M, за которым следуют **параметры**, которые должны быть переданы программе CP/M.

**cpm Имя\_программы Параметр ...**

То есть, это выглядит как командная строка CP/M с "cpm" перед ней. Например, для отладки test.com с использованием zsid.cpm в текущем каталоге команда выполните следующие действия:

**cpm zsid test.com**

Вы также можете запускать программы CP/M в других каталогах.

**cpm b:\cpm\turbo**

и когда нчего не будет указано кроме

**cpm**

отображается имя и информация об использовании.

Если ваш процессор V30, для программ, которые явно используют только инструкции 8080 попробуйте указать расширение .cpv. Используя режим эмуляции 8080 процессора V30, скорость выполнения можно увеличить в 4-8 раз. Даже если указано расширение и .cpv, и если процессор 80186 или выше используется программный симулятор.

Следующие расширения могут использоваться в качестве расширений программы CP/M.

**.cpm** Обычная программа CP/M

**.cpv** Используется режим эмуляции V30 8080, если это возможно

**.com** Ответ 122h (MP/M) с получением версии (BDOS № 12)

Расширение .com было добавлено в v0.7 в основном для возможности запуска HI-TECH C.

Стандартная библиотека времени выполнения HI-TECH C получает информацию о версии и не закрывает открытый для чтения FCB, если это не MP/M или CCP/M. По этой причине FCB тратятся впустую, а файлы, которые были открыты и прочитаны один раз, не могут быть перезаписаны в WinNT, 2K, XP и т. д. (По этой причине драйвер компилятора HI-TECH C остановится на полпути.) Чтобы избежать этого явления, драйверу ложно сообщается, что программа выполняется в MP/M, если используется расширение.com. (Функции MP/M не поддерживаются)

Для самого компилятора HI-TECH C и программ, скомпилированных с помощью HI-TECH C, оставьте расширение .com. Часто в некоторых случаях бывает полезно оставить его как .com. Однако, будьте осторожны, чтобы не запустить его по ошибке в MS-DOS.

Большинство программ должны работать с .com, но если они не работают или работает неправильно из-за странных вызовов BDOS, следует использовать расширение .cpm.

**3. Устранение неисправностей**

* Файл не может быть открыт программой CP/M

Когда запущена программа CP/M, может появиться сообщение типа "FCB не может быть использован", и файл может не открыться. Это связано с низким максимумом FCB на стороне MS-DOS. (По умолчанию 4)

Максимальное значение FCB указывается с помощью команды fcbs в CONFIG.SYS. Например, укажите fcbs = 16. 16 довольно много, но это мера против программы, которая впустую расходует FCB.

Для вступления изменений в силу требуется перезагрузка системы. Для Windows NT, 2000, XP измените %SystemRoot%\System32\config.nt вместо CONFIG.SYS. В этом случае перезапуск не требуется. Выйдите из командной строки, снова откройте ее и запустите программу DOS.

* В командной строке NT виртуальная машина DOS может быть не в состоянии захватить файл и освободить его, если программа DOS завершает работу ненормально.

В этом случае запустите диспетчер задач с помощью Ctrl-Alt-Del и принудительно завершите NTVDM.EXE.

* Отображение cpm.exe и других программ DOS исчезает в Windows NT, 2000, XP.

Даже если программа DOS выполняется нормально в командной строке NT, экран может не отображаться.

В этом случае оно, кажется, восстанавливается после того, как программа DOS находится в состоянии ожидания для ввода. Например, попробуйте выйти при помощи exit, когда появится приглашение xccp cpm.

**4. Эмулятор HD64180**

Эта программа поддерживает расширенные инструкции HD64180 в дополнение к инструкциям Z80. Однако инструкция ввода-вывода HALT SLP не поддерживается, и если вы попытаетесь выполнить ее, она завершится с сообщением

**undefined op-code executed xxxx**

IM0 IM1 IM2 EI DI - это просто NOP, а RETI RETN работают так же, как RET.

RETI RETN работает так же, как RET.

В Z80 и HD64180 регистр R увеличивается с каждой выборкой кода OP, но в этой программе это реально не выполняется. Вместо этого инструкция LD A,R генерируют случайные числа, которые заполняют регистр R. Поскольку начальное значение регистра R определяется при запуске, каждый раз это будет разная последовательность случайных чисел.

Флаги H и N не корректно работают с неарифметическими инструкциями, но я думаю, что это не вызовет проблем. Однако инструкция AND очищает эти флаги для тех, кто пишет программы, подобные приведенной ниже.

AND 0FH

DAA

ADD A,F0H

ADC A,40H

Начиная с версии 0.6, также поддерживаются следующие общеизвестные недокументированные инструкции Z80.

LD IXH,r DD 60 ～ DD 67 (01100xxx)

LD IXL,r DD 68 ～ DD 6F (01101xxx)

LD IYH,r FD 60 ～ FD 67 (01100xxx)

LD IYL,r FD 68 ～ FD 6F (01101xxx)

LD r,IXH DD 44 ～ DD 7C (01xxx100)

LD r,IXL DD 45 ～ DD 7D (01xxx101)

LD r,IYH FD 44 ～ FD 7C (01xxx100)

LD r,IYL FD 45 ～ FD 7D (01xxx101)

op A,IXH DD 84 ～ DD BC (10xxx100)

op A,IXL DD 85 ～ DD BD (10xxx101)

op A,IYH FD 84 ～ FD BC (10xxx100)

op A,IYL FD 85 ～ FD BD (10xxx101)

r= B C D E A op= ADD ADC SUB SBC AND XOR OR CP

ххх 000 001 010 011 111 ххх 000 001 010 011 100 101 110 111

**5. Режим эмуляции 8080 в V30**

В режиме эмуляции 8080 векторы INT 0F1h 0F2h 0F3h используются для запуска программы, вызова BDOS и вызова BIOS, соответственно. Если эти векторы INT уже используются, перепишите строку

**V$START = 0f1h**

в cpm.asm соответствующим образом и соберите заново, чтобы использовать другой вектор INT.

Обратите внимание, что эти векторы INT не сбрасываются, если не используется режим эмуляции 8080.

В эмуляторе начиная с v0.5 и выше измененный вектор INT восстанавливается после завершения программы.

**6. Вызовы BDOS**

Вызов BDOS - это просто соответствующий системный вызов MS-DOS. Для некоторых мы скорректировали возвращаемое значение.

Следующие вызовы BDOS не поддерживаются MS-DOS, выдавая сообщение

**Unsupported BDOS call: xx .**

Регистры HL, A возвращают 0. То же самое верно для несуществующих вызовов BDOS.

**№ 7 Получить байт ввода/вывода**

**№ 8 Установить байта ввода/вывода**

**№ 24 Получить вектор используемых дисков.**

**№ 28 Защита диска от записи.**

**№ 29 Получить вектор дисков только для чтения**

**№ 30 Установка атрибута файла.**

Кроме того, следующие вызовы BDOS не поддерживаются MS-DOS, но будут соответствующим образом обмануты.

**№ 12 Получение номера версии**

Возвращает HL = 122h. Это несуществующий код MP/M 2.2.

(необходимо для HI-TECH C)

**№ 27 Извлечение вектора распределения.**

Возвращает адрес фиктивного вектора размещения.

**№ 31 Извлечение блока параметров диска.**

Получает оставшееся дисковое пространство, создает фиктивный блок параметров диска и вектор распределения и возвращает адрес блока параметров диска.

**№ 32 Получение/установка кода пользователя**

не оказывает никакого влияния, но может извлечь установленный код пользователя.

**7. Вызовы BIOS**

Поддерживаются только следующие вызовы BIOS.

**00h Холодный старт**

**03h Перезагрузка**

Завершит программу в обычном режиме.

**06h Получить статус консоли**

IOCTL (4406h) проверяет, готов ли стандартный ввод. Используйте в состоянии "break<OFF>".

**09h Ввод символа с консоли**

Используется прямой консольный ввод/вывод (07h).

**0ch Вывод символа на принтер**

Используется прямой консольный ввод /вывод (06h).

**0fh Вывод на перфоратор**

Используется вывод на принтер (05h). Проверяется ^C.

**12h Используется вывод AUX (04h). Проверяется ^C.**

**15h Ввод со считывателя**

Используется вход AUX (03h). Проверяется ^C.

**8. Сборка**

Для сборки требуется OPTASM. Смотри MAKEFILE.

Собирая без опции /Dundoc, вы можете предотвратить выполнение недокументированных инструкций Z80.

**9. Библиография**

"Создание двойного эмулятора ACE" Junhiko Kanai

Процессор 1986 года марта c.37-c.48

"Построение нестандартного CP/M" Такахара, Катагири, Ямамото

Интерфейс 1982 декабрь выпуск c.162-c.219

"HD64180 Инструкция карманная книжка" Hitachi

"HD64180 Руководство по применению" Hitachi, Ltd.

"Справочник программиста MS-DOS 3.1" NEC

**10. Получение программ CP/M**

Разработка коммерческих программ CP/M давно прекращена и они больше не доступны для продажи у розничных продавцов. Тем не менее, бесплатные программы можно скачать со следующих сайтов.

*Неофициальный веб-сайт CP/M*

<http://www.cpm.z80.de/>

Основной блок CP/M, CBASIC, COBOL-80, различные FORTH, MAC, PASCAL MT+, PL/I-80, PL/M, ZSID, BDS-C, Modula-2, SMALL C и т. д.

*BDS C: компилятор 8080/Z80 C*

<http://www.bdsoft.com/resources/bdsc.html>

*HI-TECH Z80 CP/M C Compiler (версия 3.09)*

<http://www.retroarchive.org/cpm/cdrom/SIMTEL/HITECH-C/>

(Похоже, распространение закончилось)

*Walnut Creek CP/M CD-ROM online*

<http://www.retroarchive.org/cpm/cdrom/>

**11. XCCP**

XCCP - это расширенный консольный командный процессор, созданный O.I. (имя скрыто по желанию). Он работает вместо CCP и добавляет различные функции.. Программа xccp.cpm - это патч XCCP с расширением командного файла по умолчанию .cpm, который поставляется вместе с cpm.exe с разрешения OI.

Использование

**cpm xccp**

**Функции**

* Считать командную строку из консоли, загрузить и запустить программу
* Выполнить внутренние команды (описаны ниже)
* Ввод нескольких команд в одной строке, разделенных символом "!"
* Выполнить файл SUBMIT ▲

**Внутренние команды**

d: Изменить текущий диск

DIR d: Показать каталог

TYPE file Отобразить файл, можно использовать подстановочные знаки ★

REN file = file Переименовать файл

ERA file Удалить файл

SAVE n file Сохранить содержимое TPA в файл

USER n Изменить номер пользователя ▲

CONT Повторно выполнить предыдущую команду (Запуск без загрузки программы) ★

LGON file Начать запись вывода на экран в файл ★

LOFF Прекратить запись вывода на экран в файл ★

XSUB Встроенный XSUB.COM ★ ▲

EXIT Выход из XCCP ★

★: расширенная функциональность для CCP

▲: функции, которые нельзя использовать с cpm.exe

Функция SUBMIT не может использоваться в текущей версии cpm.exe. Это потому, что XCCP переписывает внутреннюю часть FCB для произвольного доступа и изменения размера файла.

Мистер O.I., похоже, избавился от всех материалов и программам, связанных с CP/M, и исходный текст программы XCCP потерян. Поэтому мы не можем отвечать на вопросы и запросы, связанные с XCCP. Он не будет обновлен.

**12. Условия распространения**

Использование и распространение этой программы разрешается при следующих условиях:

* Бесплатно

Эта программа бесплатна. Никакие другие расходы или сборы не должны взиматься за распространение.

* Отсутствие гарантий

Автор (Кейджи Мураками) не дает никаких гарантий для этой программы.

* Недопустимо создавать помехи использованию и распространению третьими лицами

Пока третье лицо соблюдает эти условия, использование и распространение им не ограничено.

Кроме того, следующие условия разрешают изменения этой программы и распространение модификаций.

* Указание человека, который внес исправление, и часть, подлежащая исправлению.
* Распространение на тех же условиях, что и оригинал.

**13. Содержание распространения**

CPM07.TXT Этот файл

CPM.EXE Исполняемый файл

CPM.ASM Исходный файл (эмулятор CP/M)

EM180.ASM Исходный файл (эмулятор HD64180)

MSDOS.MAC Включаемый файл

MAKEFILE Файл сборки

XCCP.CPM Исполняемый файл процессора расширенной консоли

**14. Благодарность**

Спасибо г-ну O.I. за согласие включить XCCP.

Я также хотел бы поблагодарить г-на Пнори и г-на Цудзи за то, что они не только отправили сообщения об ошибках, но и помогли с отладкой.

Мы получили ценную информацию о HI-TECH C на сайте "Tatsu's MSX Researches)". Спасибо Тацу и Кену Хасэгавага. Кроме того, 2-канала "Старый ПК" и "Кстати об ОС, вот она! CP/M", предоставили информацию о работе MBASIC. Спасибо ветке резидентов..

Г-н Кейичи Миямото прислал сообщение об ошибке на начальном значении SP, а г-н Нозоми Фудзита отправил сообщение об ошибке с флагами H и N. Спасибо обоим.

Затем человек, представивший отчет об ошибке в 1996 году, забыл указать свое имя. Извините. Спасибо.

**15. История**

1989/04/22 **v0.1** Выпущена

1989/05/03 **v0.2** Исправлены ошибки командной строки

1989/08/19 **v0.3** Добавлена встроенная версия эмуляции 8080 процессором V30

Сжатие размера файла EXE

1989/12/19 **v0.4** Исправлена ошибка при использовании процессора 8086

регистр R получает случайные начальные значения

Исправлена ошибка при поиске файла

1994/09/25 **v0.5** Исправлены ошибки для CPIR, CPDR, LDIR, LDDR (найдены Пнори)

Исправлена ошибка, установки флага V

Соответствие английскому DOS (сообщения переведены на английский)

Изменено восстановление векторов при использовании режима эмуляции 8080 процессором V30

1994/10/23 **v0.6** Исправление ошибки SET, RES (IX, IY) (обнаружено г-ном Цудзи)

Поддержка недокументированных инструкций Z80, которые обрабатывают верхний и нижний значения IX и IY как 8-битные регистры

2004/07/21 **v0.7** Улучшенный метод обнаружения V30

Исправления ошибок для кодов операций (IXL, IY, IXL, IX)

Повышение скорости (486 и более поздние версии)

Изменение диска A: на C: если диск съемный

Выполнение продолжить выполнение с неопределенным вызовом BDOS (требуется для HI-TECH C)

Исправление, делающее расширение .COM исполняемым

Функция BDOS № 12 возвращает 122h (MP/M 2.2) и выше

Улучшение раздела ввода BIOS (требуется для MBASIC）

Исправлены орфографические ошибки на дисплее

2009/04/16 Изменение первоначального значения SP

2012/05/31 **v0.8** Улучшена работа флагов H, N

**16. Контактная информация автора**

NIFTY-Serve PEH00316 Mu

keiji-mu@nifty.com

Keiji Murakami {Кейджи Мураками}

**Программа CP/M-80 EXEcutor для Win32 V0.4**

cpm.exe

Copyright (C) 2005-2012 Keiji Murakami {Кейджи Мураками}

**0. Введение**

Я пытался сделать версию Win32 к 15-летию cpm.exe версии MS-DOS.

Следующие функции унаследованы от версии MS-DOS:

1. Выполнение программы CP/M-80 непосредственно из командной строки;
2. Программа CP/M-80 может быть выполнена в пакетном файле или Makefile;
3. Работа в среде CP/M-80 с подключенным резидентным CCP;
4. Поддержка HI-TECH C;
5. Поддержка MBASIC.

Кроме того, были добавлены следующие функции:

1. Поддержка Win32;
2. Улучшенная обработка файлов с помощью собственной эмуляции BDOS;
3. Указание пути поиска для программ CP/M, используя переменную окружения CPMPATH;
4. Перенаправление в файл устройства CP/M по переменной окружения;
5. Назначение каталога для диска CP/M по переменной окружения;
6. Автоматическое назначение привода по аргументу каталога;
7. Загрузка процессора и регулирование скорости при вводе с консоли во время ожидания;
8. Эмуляция терминала ANSI + SOLOC (Televideo);
9. Автоматически генерируемое сочетание клавиш Ctrl-Z при перенаправлении ввода и автоматическом завершении программы;
10. Поддержка кода завершения в стиле HI-TECH C;
11. Поддержка функции завершения ошибок при удалении A:$$$.SUB (обнаружение ошибок для BDS-C).

**1. Рабочая среда**

**CPU**: 80386 совместимый

**ОС**: Windows 95, 98, Me, NT, 2000, XP, Vista, 7 (рекомендуется NT или более поздняя версия) (Также работает на Linux + Wine)

**2. Эмулируемая среда**

**CPU**: Z80 (включая инструкции HD64180 и некоторые недокументированные инструкции, кроме инструкций ввода-вывода)

**ОС**: Эквивалент CP/M-80 Ver2.2 (включая некоторые вызовы BDOS Ver 3.0)

**Карта памяти:**

0000h – 00FFh Нулевая страница

0100h - FDFFh TPA

FE00h – FEFFh BDOS + рабочая зона

FF00h – FFFFh BIOS + рабочая область

**Назначение диска:**

**A**: Каталог программ CP/M-80 (диск, выбранный с опцией **-a**)

**B**: Текущий каталог (диск выбранный по умолчанию)

**C**: Первый каталог, заданный первым и вторым параметрами (автоматическое назначение диска) или каталог, заданный переменной среды **CPMC**

**D**: Второй каталог, заданный первым и вторым параметрами (автоматическое назначение диска) или каталог, заданный переменной среды **CPMD**

**E**: Каталог, указанный в переменной окружения **CPME**

...

**P**: Каталог, указанный в переменной среды **CPMP**

**Назначение устройства:**

**LST:** Файл или устройство, указанное в переменной окружения **CPMLST**

По умолчанию "PRN:"

**RDR:** Файл или устройство, указанное в переменной среды **CPMRDR**

Файл или устройство, указанное в переменной окружения **CPMAUX**

По умолчанию "AUX:"

**PUN:** Файл или устройство, указанное в переменной среды **CPMPUN**

Файл или устройство, указанное в переменной окружения **CPMAUX**

По умолчанию "AUX:"

**2. Быстрый старт**

Сначала перенесите программу CP/M-80 в Windows. В настоящее время, если задано расширение ".cpm" вместо".com", оно не будет ошибочно принято за исполняемый файл Windows.

Для запуска программы CP/M-80 первым аргументом cpm.exe укажите **имя программы** CP/M-80, за которым следуют **параметры**, которые должны быть переданы программе CP/M-80.

**cpm Имя\_программы Параметр ...**

То есть, это выглядит как командная строка CP/M с "**cpm**" перед ней.

Например, для отладки test.com с использованием zsid.cpm в текущем каталоге выглядит следующим образом:

**cpm zsid test.com**

Вы также можете запускать программы CP/M, расположенные в других каталогах.

**cpm b:\cpm\turbo**

и когда нчего не будет указано кроме

**cpm**

отображается имя и информация об использовании.

Следующие расширения могут использоваться в качестве расширений программы CP/M.

.cpm Обычная программа CP/M

.com Ответ 122h (MP/M) с получением версии (BDOS № 12)

Вы также можете выполнять файлы с любым расширением, указав расширение.

**cpm prog.z80**

**3. Использование**

Использование: **CPM [-hxapdC] [-w[0-9]] команда arg1 arg2 ...**

**-h** вернуть код выхода HI-TECH C

**-x** вернуть ERROR, если удален файл A:$$$.SUB

**-a** выбрать A: (каталог c программами)

**-p** пауза перед выходом

**-d** отключить автоматическое назначение диска

**-C** аргументы в верхнем регистре

-**w**[**0**-**9**] ожидание проверки состояния консоли (макс. 9)

**Описание аргументов командной строки**

**команда** Укажите имя файла программы CP/M-80. Если расширение не указано, к имени файла будет добавлено .cpm или .com. Если диск и каталог опущены, поиск файлов осуществляется из каталога, указанного в переменной среды **CPMPATH**, в дополнение к текущему каталогу текущего диска.

**arg** <**n**> Укажите параметры, которые будут переданы в программу CP/M-80.

если параметр -d не указан, первые два параметра автоматически назначаются приводу. Если эти параметры содержат реальные спецификации каталога, система автоматически назначает их диску C:, D: и заменяет спецификацию каталога параметра на букву диска, которой он назначен.

**Описание опций**

**-h** Возвращает код выхода в стиле HI-TECH C.

2 байта расположенные по адресу 0x80 являются кодом завершения. Эта опция является эксклюзивной для -x и вступит в силу после ее указания.

**-x** Устанавливает код выхода равным 2, когда вызов удаления BDOS выполнен для A:$$$.SUB. В противном случае возвращает 0.

BDS-C и т.д. выдает вышеуказанный вызов BDOS, чтобы прервать отправку при возникновении ошибки, в зависимости от опции. Указав опцию -x, наличие или отсутствие ошибки, возникшей в таком приложении, может быть отражено в коде выхода.

Она является эксклюзивной для -h и вступит в силу после ее указания.

**-a** Выбор диска A:

CPM.EXE назначает каталог, содержащий программные файлы CP/M-80, диску A: и текущий каталог диску B: и начинает выполнение программы обычно с выбранным диском B:. Параметр -a исправляет это поведение и запускает программу с выбранным диском A:.

Используйте эту опцию для приложений, которые ожидают файлы (библиотеки и оверлеи), необходимые для запуска, на выбранном диске. (Например, для BDS-C по умолчанию)

**-p** Пауза до нажатия любой клавиши перед выходом.

Когда программа CP/M-80 выполняется двойным щелчком по файловой ссылке и т.п., консоль закрывается сразу после выхода, но если присутствует эта опция, можно прочитать конец отображения вывода программы CP/M-80.

**-d** Подавляет функцию автоматического назначения привода на основе параметров CP/M-80.

Обычно, когда первый и второй параметры (arg1, arg2), передаваемые программе CP/M-80, содержат в себе спецификации существующих каталогов, CPM.EXE автоматически сопоставляет каталоги с CPM дисками C: и D:, заменяя спецификацию каталога параметра на букву диска, которой он назначен.

Эта опция подавляет функцию автоматического назначения диска, описанную выше, и передает параметры в программу CP/M-80 как есть.

**-C** Преобразует параметры командной строки в верхний регистр.

Исходный CCP использует параметры командной строки с прописной буквы и сохраняет их начиная с адреса 80h или FCB по умолчанию, что ближе к этому поведению.

Ее можно использовать для программ, которые работают со сбоями, если параметры не прописные, например, zlink.

**-w**[**0**-**9**] При проверке состояния консоли вставка ожидания вызова BDOS.

Числа 0-9 являются весами, чем больше значение, тем выполнение программы медленнее. Если число не указано, предполагается, что оно равно 1.

Ее можно использовать для регулировки скорости, если интерактивная игра слишком быстрая. Она также может быть использована для предотвращения 100% загрузки процессора во время выполнения интерактивной программы.

**Переменные окружения**

**CPMPATH** Определяет путь поиска для программных файлов CP/M-80.

Он имеет тот же формат, что и переменная окружения PATH.

**CPMC** - **CPMP** Определяет каталог, который будет назначен дискам C: - P: соответственно. Однако **CPMC** и CPMD игнорируются, когда выполняется автоматическое назначение привода по параметру.

**CPMLST** CP/M-80 LST: определяет файл или устройство, которое будет назначено устройству вывода. Значением по умолчанию является PRN:.

**CPMRDR** CP/M-80 RDR: определяет файл или устройство, которое будет назначено устройству ввода. Если эта переменная среды не существует, будет использоваться переменная среды CPMAUX.

**CPMPUN** CP/M-80 PUN: определяет файл или устройство, которое будет назначено устройству вывода. Если эта переменная среды не существует, будет использоваться переменная среды CPMAUX.

**CPMAUX** CPMRDR, CPMPUN определяет файл или устройство, которое будет назначено для CP/M-80 RDR: PUN: устройство, когда нет переменной среды. Файл или устройство, указанные в этой переменной среды, открываются в режиме чтения/записи, считываются через устройство RDR: и записываются через устройство PUN:.

Значением по умолчанию является AUX:.

**4. Эмулятор Z80 (HD64180)**

Эта программа поддерживает расширенные инструкции HD64180 MLT и TST в дополнение к инструкциям Z80. Однако инструкция ввода-вывода HALT SLP не поддерживается и попытки ее выполнения завершаются ошибкой:

**ERROR: Illegal op-code executed. xxxx**

или

**ERROR: Halted. xx**

IM0 IM1 IM2 EI DI - это просто NOP, а RETI RETN работают так же, как RET.

RETI RETN работает так же, как RET.

В Z80 и HD64180 регистр R увеличивается с каждой выборкой кода OP, но в этой программе это реально не выполняется. Вместо этого инструкция LD A,R генерируют случайные числа, которые заполняют регистр R. Поскольку начальное значение регистра R определяется при запуске, каждый раз это будет разная последовательность случайных чисел.

Флаги **H** и **N** не корректно работают с неарифметическими инструкциями, но я думаю, что это не вызовет проблем.

Также поддерживаются следующие общеизвестные недокументированные инструкции Z80.

LD IXH,r DD 60 ～ DD 67 (01100xxx)

LD IXL,r DD 68 ～ DD 6F (01101xxx)

LD IYH,r FD 60 ～ FD 67 (01100xxx)

LD IYL,r FD 68 ～ FD 6F (01101xxx)

LD r,IXH DD 44 ～ DD 7C (01xxx100)

LD r,IXL DD 45 ～ DD 7D (01xxx101)

LD r,IYH FD 44 ～ FD 7C (01xxx100)

LD r,IYL FD 45 ～ FD 7D (01xxx101)

op A,IXH DD 84 ～ DD BC (10xxx100)

op A,IXL DD 85 ～ DD BD (10xxx101)

op A,IYH FD 84 ～ FD BC (10xxx100)

op A,IYL FD 85 ～ FD BD (10xxx101)

r= B C D E A op= ADD ADC SUB SBC AND XOR OR CP

ххх 000 001 010 011 111 ххх 000 001 010 011 100 101 110 111

Кроме того, были расширены инструкции по вызову функций BDOS и BIOS. Эти коды соответствуют инструкциям расширения V20/30 в режиме эмуляции 8080.

ED ED 00 CALLN 0 Вызов BDOS

ED ED xx CALLN xx Вызов BIOS

ED FD RETEM Конец эмуляции

**6. Вызовы BDOS**

Функции BDOS реализуются программами на C на стороне Win32. Код на стороне Z80 (HD64180) содержит только - CALLN 0; RET.

Следующие вызовы BDOS не поддерживаются: ни чего не делают, выдавая сообщение

**Unsupported BDOS call: xx**

Регистры HL, A возвращают 0. То же самое верно для несуществующих вызовов BDOS.

**№ 7 Получить байт ввода/вывода**

**№ 8 Установить байт ввода/вывода**

**№ 28 Защита диска от записи.**

**№ 29 Получить вектор дисков только для чтения**

**№ 30 Установка атрибута файла.**

Кроме того, следующие вызовы BDOS выполняются следующим образом.

**№ 12 Получить номер версии**

Возвращает номер версии в регистрах A и HL.

Старший байт номера версии изменяется в зависимости от расширения исполняемой программы. При маскировке MP/M библиотека времени выполнения HI-TECH C 80 закроет файл, даже когда он открыт для чтения.

|  |  |  |  |
| --- | --- | --- | --- |
| Расширение | A | HL | Описание |
| .cpm | 22h | 0022h | CP/M-80 2.2 |
| другие | 22h | 0122h | MP/M 2.2 (отсутствует) |

**№ 27 Получить вектор распределения**

Возвращает адрес фиктивного вектора размещения.

**№ 31 Получить блок параметров диска**

Получает оставшуюся емкость диска, составляет фиктивный блок параметров диска и вектор распределения, а также возвращает адрес блока параметров диска.

**№ 32 Получить / установить код пользователя**

Несмотря на отсутствие эффекта, но вы можете получить код пользователя, который вы установили..

Также поддерживаются следующие расширенные вызовы BDOS CP/M-80 3.0.

Они используются библиотекой времени выполнения HI-TECH C 80.

**№ 102 Получить время файла**

**№ 105 Получить текущее время**

**7. Вызовы BIOS**

Поддерживаются только следующие вызовы BIOS.

**00h Холодный старт**

**03h Перезагрузка**

Завершит программу в обычном режиме.

**06h Получить статус консоли**

Проверка возможности ввода стандартного ввода.

Если ввод невозможен и указана опция -w, сон будет выполняться с частотой и длительностью в соответствии с указанным значением.

**09h Ввод символа с консоли**

Получает один символ из стандартного ввода.

Если стандартный ввод является консолью, также выполняется преобразование кода символа.

Кроме того, если стандартный ввод является прямым и вызывается после достижения конца файла, то в качестве входного символа используется до 16 раз Ctrl-Z (1A) .Если он вызывается более 16 раз, то автоматически завершает работу программы. В это время код выхода будет равен 3.

**0ch Вывод символа на консоль**

Выводит один символ на стандартный вывод.

Если стандартным выводом является консоль, интерпретируйте коды управления экраном ANSI, SOLOC.

**0fh Вывод на принтер**

Выводит один символ в файл или устройство, указанное в переменной окружения CPMLST. Если не указано, вывести в PRN:.

**12h Вывод на перфоратор**

Выводит один символ в файл или устройство, заданное переменной окружения CPMPUN. Если CPMPUN не указана, выводит в файл или устройство, указанное переменной окружения CPMAUX, в режим чтения-записи. Если CPMAUX также не указан, то выводится в режиме чтения-записи на AUX:.

**15h Ввод со считывателя бумажной ленты**

Получает один символ из файла или устройства, указанного переменной окружения CPMRDR. Если CPMRDR не указана, ввод осуществляется в режиме чтения-записи из файла или устройства, заданного переменной окружения CPMAUX. Если CPMAUX также не определена, ввод осуществляется в режиме чтения-записи из AUX:.

Если происходит вызов неподдерживаемого вызова BIOS, отображается сообщение

**ERROR: Unsupported BIOS call: xx**

и cpm.exe завершается.

**8. Эмуляция терминала**

Если стандартный вывод не перенаправляется на консоль, cpm.exe интерпретирует коды управления экраном для эмуляции терминалов ANSI и SOLOC.

**Коды управления экраном ANSI**

**ESC [ y ; x H** Перемещает курсор в абсолютное положение (x, y)

**ESC [ y ; x f** - // -

**ESC [ dy A** Перемещает курсор вверх на dy

**ESC [ dy B** Перемещает курсор вниз на dy

**ESC [ dx C** Перемещает курсор влево на dx

**ESC [ dx D** Перемещает курсор вправо на dx

**ESC [ J** Очищает экран от текущей строки до нижней границы экрана.

**ESC [ 1 J** Очищает экран от текущей позиции до верхней границы экрана

**ESC [ 2 J** Стирает весь экран

**ESC [ K** Стирает текст от курсора до конца строки

**ESC [ 1 K** Стирает текст от начала строки до курсора

**ESC [ 2 K** Стирает всю строку

**ESC [ dy M** Прокручивает экран на dy строк вверх

**ESC [ dy L** Прокручивает экран на dy строк вниз

**ESC [ u** Восстанавливает позицию курсора

**ESC [ s** Сохраняет позицию курсора

**ESC [ m** Очищает атрибуты символа

**ESC [ ... m** Устанавливает атрибуты символа

**Код атрибута символа**

**1** Повышенная яркость

**4** Подчеркивание

**7** Инверсия

**16**-**23** Цвет текста (черный, зеленый, красный, желтый, синий, голубой, пурпурный, белый)

**30**-**37** Цвет шрифта (черный, зеленый, синий, голубой, красный, желтый, пурпурный, белый)

**40**-**47** Цвет фона (черный, зеленый, синий, голубой, красный, желтый, пурпурный, белый)

**Коды управления экраном SOLOC**

**CLS (1Ah)** Стирает экран

**ESC \*** Стирает экран

**ESC = y x** Перемещает курсор на (x-20h, y-20h)

**ESC (** Повышенная яркость символов

**ESC )** Отмена выделения символов

**ESC t** Стирает текст от курсора до конца строки

**ESC T** Стирает текст от курсора до конца строки

**ESC y** Стирает текст от курсора до конца страницы

**ESC Y** Стирает текст от курсора до конца страницы

**ESC E** Вставка строки

**ESC R** Удаление строки

**ESC D** Прокрутка по верхнему краю

**ESC M** Прокрутка по нижнему краю курсора

Если стандартный ввод не перенаправляется на консоль, cpm.exe выполняет преобразование кода клавиши, и при нажатии функциональной клавиши вводит следующий код:

UP 05h (^E)

DOWN 18h (^X)

LEFT 13h (^S)

RIGHT 04h (^D)

INS 16h (^V)

DEL 7Fh

PgUP 12h (^R)

PgDOWN 03h (^C)

^UP 17h (^W)

^DONW 1ah (^Z)

^LEFT 01h (^A)

^RIGHT 06h (^F)

**8. Компиляция**

Для компиляции cpm.exe требуется Borland C++ Builder. Для ассемблирования файлов em180.asm, z80op.txt требуется jperl.exe. Смотрите src\Makefile.

Эмулятор v0.4 или новее поддерживает сборку VC++. Закомментируйте определение для BCC в начале файла src\Makefile и, включите определение для VC++, чтобы использовать его.

Также для сборки прилагаемых программ RCCP.COM, 4GCLOCK.COM и компиляции INFO0P.COM необходим HI-TECH C 80. Смотрите utl\Makefile.

**10. Получение программ CP/M**

Разработка коммерческих программ для CP/M давно прекращена, и их трудно получить у дилеров. Тем не менее, бесплатные программы можно скачать со следующих сайтов.

Неофициальный веб-сайт CP/M

<http://www.cpm.z80.de/>

Код CP/M, CBASIC, COBOL-80, различные FORTH, MAC, PASCAL MT+, PL/I 80, PL/M, ZSID, BDS-C, Modula-2, SMALL C и т. д.

BDS C: компилятор C 8080/Z80

<http://www.bdsoft.com/resources/bdsc.html>

Компилятор HI-TECH C Z80 для CP/M (версия 3.09)

<http://www.retroarchive.org/cpm/cdrom/SIMTEL/HITECH-C/>

(Похоже, распространение закончилось)

Walnut Creek CP/M CD-ROM online

<http://www.retroarchive.org/cpm/cdrom/>

**11. RCCP**

RCCP.COM является резидентным командным процессором консоли. Он позволяет работать в среде CP/M. RCCP.COM был создан с использованием ассемблера ZAS, включенного в HI-TECH C 80.

**Использование**

**cpm rccp**

**Функции**

* Чтение командной строки из консоли и загрузка/выполнение программы (в качестве программы признает не только .com, но и .cpm) ★
* Выполнение внутренних команд (будут описаны позже)
* Выполнить файл SUBMIT $$$.SUB (Вы можете выполнить $$$.SUB на диске B: а также на диске A:) ★

**Внутренние команды**

d: Изменить текущий диск

DIR d: Показать каталог

TYPE file Отображение файла

REN file = file Переименовать файл

ERA file Удалить файл

SAVE n file Сохранить содержимое TPA в файл

USER n Изменить номер пользователя ▲

CONT Повторно выполнить предыдущую команду (выполнить без загрузки программы)

XSUB встроенный XSUB.COM ★

EXIT Выход из RCCP ★

★: Расширенные функции для CCP

▲: функции, которые нельзя использовать с cpm.exe

Версия CPM.EXE для MS-DOS в качестве резидентного командного процессора консоли использует XCCP.COM. Однако, при условии распространения, соответствующим GPL на этот раз, стало сложно прикрепить небольшой источник XCCP.COM. Кроме того, XCCP.COM поддерживает только расширение .cpm (первоначально .com), что затрудняет его использование в смешанной среде .cpm .com. Поэтому мы решили создать подмножество XCCP.COM с нуля и прикрепить его к исходному коду. Разница между XCCP.COM и RCCP.COM заключается в следующем.

* Отсутствует функция последовательного выполнения команд, разделенных '!'
* Отсутствует функция вывода файла для вывода на экран файла LGON, команда LOFF
* В команде TYPE нельзя использовать подстановочные символы

Вы все еще можете использовать XCCP.COM с cpm.exe версии Win32. Вы также можете выполнить SUBMIT файлы, которые были невозможны в версии DOS. Однако сама команда SUBMIT должна быть подлинной.

В v0.1 RCCP и XCCP прерывались вводом Ctrl-C, но в v0.2 и более поздних версиях теперь можно вернуться в RCCP и XCCP с помощью Ctrl-C при выполнении вызова BDOS/BIOS. Обратите внимание, что принудительное завершение с помощью Ctrl-Break все еще действует.

Действие при вводе Ctrl-C:

| Выполнение кода Z80 | Вызов BDOS/BIOS

------ + ------------------------- + ------------------------

v0.1 | cpm.exe завершается сам | cpm.exe завершается сам

v0.2 | Игнорирование | Возврат в RCCP/XCCP

**12. Другие вложенные файлы**

**4GCLOCK.COM**

Программа тестирования производительности, которая потребляет около 4000 000 000 часов на Z80. Она ничего не делает, но потребляет время. Пожалуйста, измерьте время работы. Оно должно составить 1000 секунд на Z80 4 МГц.

В моей среде результат выглядит следующим образом:

Athlon 64 X2 5400+ (2,8 ГГц) Win7 .. 1,20 секунды (эквивалент Z80 3,3 ГГц)

Core2 Duo E8400 (3,0 ГГц) Win7 .. 0,81 секунды (эквивалент Z80 4,9 ГГц)

**patch$exec.txt**

Этот файл используется для исправления интерпретатора команд HI-TECH C 80 $EXEC.COM. Требуется ZSID и RCCP.

Программа, скомпилированная HI-TECH C 80, может вернуть код завершения, выполненная **cpm.exe** с опцией **-h**. Это так же относится к препроцессору HI-TECH C 80 (CPP.COM)" и программам компилятора (CGEN.COM) и компоновщику (LINK.COM).

К сожалению, драйвер компилятора (C.COM) всегда возвращает 0 успеха. Поэтому, если вы говорите **cpm -h c prog.c** в Makefile, утилита make не может быть прервана, даже если произойдет ошибка компиляции.

Поэтому примените патч к $EXEC.COM, который вызывается из C.COM и фактически выполняет препроцессор, тело компилятора, компоновщик и т. д., чтобы область кода возврата не очищалась. В частности, установите третий байт из байта "1BBh" $EXEC.COM в 00h (NOP).

При условии, что $EXEC.COM и patch$exec.txt находятся в текущем каталоге, а ZSID.COM и RCCP.COM находятся в текущем каталоге или каталоге, указанном в CPMPATH, Перенаправьте path$exec.txt на стандартный ввод и запустите rccp следующим образом:

**cpm rccp < patch$exec.txt**

Это исправит $EXEC.COM в текущем каталоге. Оригинал остается с именем $$EXEC.COM.

**INFO0P.COM**

Отображает следующее содержимое нулевой страницы:

Адрес записи теплой загрузки BIOS

Расположение BIOS

Адрес входа BDOS

Размер TPA

FCB по умолчанию

Параметры командной строки

Вы можете проверить состояние выделения памяти и результаты автоматического назначения привода.

info0p.c может быть скомпилирована HI-TECH C 80.

**13. Условия распространения**

Соответствует GPL2.

**14. Содержание распространения**

cpm32\_04.txt Этот файл

cpm.exe Исполняемый файл

RCCP.COM Исполняемые файлы командного процессора резидентной консоли

4GCLOCK.COM Исполняемая программа часов 4G

COPYING ОБЩАЯ ОБЩЕСТВЕННАЯ ЛИЦЕНЗИЯ GNU Версия 2

src\ Полный исходный код cpm.exe

Makefile Makefile для эмулятора

cpm.c Исходный файл (эмулятор CP/M)

em180.asm Исходный файл (эмулятор HD64180)

em180.h Заголовочный файл (эмулятор HD64180)

mkem180.pl Скрипт perl для генерации em180.asm

z80op.txt Таблица кодов операций (требуется для mkem180.pl)

mkz80op.pl Скрипт perl для генерации

cpm.rc Файл ресурсов

cpm.ico Файл иконки

utl\ Полный исходный код прилагаемой программы

Makefile Makefile для прикрепленной программы

rccp.as Источник RCCP (требуется HI-TECH C 80)

loader.as Автопереключатель (требуется HI-TECH C 80)

4Gclock.as Источник программы часов 4G (требуется HI-TECH C 80)

patch$exec.txt Данные патча $EXEC.COM (для перенаправления ввода)

info0p.c Источник INFO0P (требуется HI-TECH C 80)

**15. История**

2005/09/14 **v0.1** Выпущена

2005/09/26 Изменение внутренней структуры FCB (RC 7bit: поддержка компоновщика DRI PL/I-80)

2005/10/11 Добавление иконок

2005/12/15 **v0.2** Добавлена ловушка для установки pc в 0 с помощью Ctrl-C

2007/01/23 Удален CR в стандартном выводе (CR LF был изменен на CR CR LF)

2008/12/04 **v0.3** Исправление метода повторного использования открытого файла (совместим с TURBO PASCAL)

Поддерживает свободное место на диске 2 ТБ или более

2012/03/15 **v0.4** BDOS 36 (установить случайную запись) Исправление ошибки (поддержка UNARC)

Изменен символ стирания экрана на пробел (совместимо со средой wine console)

Поддержка сборки VC ++ / MASM

Добавлена опция -C

Изменено для использования "=:; <>" в качестве разделителя при настройке FCB по умолчанию

Улучшена работа флагов H и N в эмуляторе Z80 (64180)

Исправлена ошибка в последовательности управления экраном

Исправлена ошибка, из-за которой 7fh не вводился клавишей DEL

**16. Благодарность**

Большинство исправлений в v0.4 основаны на отчете об ошибках Nozomi Fujita и предложенных исправлениях.

Спасибо Нодзоми Фудзите.

**17. Контактная информация автора**

keiji-mu@nifty.com Keiji Murakami {Кейджи Мураками}

SCC(1) SCC(1)

**ИМЯ**

**scc** - SCC - простой компилятор C с помощью магии

**СИНТАКСИС**

**scc [-cdgkqQsWw] [-M|-E|-S] [-D def[=val]]... [-I dir]... [-L dir]...**

**[-U def]... [-l lib] [-m arch] [-o outfile] [-O level] [-t sys]**

**sourcefile...**

**ОПИСАНИЕ**

**scc** - это простой компилятор C, который берет несколько исходных файлов и компилирует их в исполняемый файл. Выходным файлом по умолчанию является a.out, который можно изменить, определив некоторый выходной файл.

**ОПЦИИ**

**-c** Не запускать компоновщик.

**-d** Вывести внутренние сообщения инструмента.

**-D define[=value]**

Определить символ для препроцессора.

**-E** Остановиться после этапа предварительной обработки, не запускать компилятор и вывести полученный в результате источник.

**-g** Генерировать отладочную информацию.

**-I** directory

Определить каталог для получения заголовочных файлов. Поиск в этом каталоге происходит до поиска в стандартных каталогах.

**-k** Сохранять временные объекты.

**-l library**

Собирать с этой биьлиотекой.

**-L directory**

Определить каталог библиотеки для разрешения зависимостей. Этот каталог используется перед стандартными путями.

**-m architecture**

Определить архитектуру для компиляции (т.е. amd64, i386...).

**-M** Вывести правило для make, описывающее зависимости основного исходного файла.

**-o outfile**

Определить имя выходного файла.

**-O level**

Определить уровень оптимизации при компиляции. (Это заглушка по соображениям совместимости.)

**-q** Не использовать QBE.

**-Q** Использовать QBE.

**-s** Удалить все таблицы символов и информацию о перемещении из результирующего исполняемого файла.

**-S** Остановиться после этапа компиляции, не ассемблировать и вывести исходный код ассемблера.

**-t system**

Определить систему для компиляции (например, linux, openbsd ...).

**-U define**

Сбросить ранее определенный символ. Обратное действие -D.

**-W** Показывать предупреждающие сообщения.

**-w** Не показывать предупреждающие сообщения (по умолчанию).

**ПЕРЕМЕННЫЕ ОКРУЖАЮЩЕЙ СРЕДЫ**

Некоторые переменные среды управляют поведением scc.

**ARCH** определяет архитектуру для компиляции (например, amd64, i386 ...).

**SYS** определяет систему для компиляции для (например, linux, openbsd ...).

**ABI** определяет двоичный интерфейс приложения для компиляции для (т. е. sysv ...).

**FORMAT** определяет формат выходного исполняемого файла.

**SCCPREFIX** определяет префикс пути, по которому scc будет искать пакет scc.

**АВТОРЫ**

Смотрите файл LICENSE для авторов.

**ЛИЦЕНЗИЯ**

Смотрите файл LICENSE для условий перераспределения.

**СМОТРИТЕ ТАКЖЕ**

scpp(1), make(1)

**ОШИБКИ**

Смотрите файл TODO в дистрибутиве.

scc-VERSION SCC(1)

MTREL.WS

Компилятор HI-TECH C 3.09, по-видимому, является одним из лучших вариантов для локальной разработки C под CP/M-80 (с Z80). Похоже, что он генерирует «разумный» код Z80, имеет разумную библиотеку времени выполнения и фактически поддерживает прототипы ANSI.

Однако, как и многие аналогичные продукты того времени, он представляет собой законченную «систему». У него есть компилятор C, Ассемблер и редактор связей. Идея состоит в том, что пользователь покупает всю систему.

Поскольку компилятор HI-TECH C (как и Whitesmiths C) работает в отдельных проходах и всегда производит ассемблер, ассемблер может быть преобразован в формат, подходящий для другой системы. Я выбрал Microsoft M80/L80. Файлы REL также используются инструментами Digital Research, такими как RMAC, что дает нам большую гибкость.

Драйвер HI-TECH C (C.COM) можно запустить так:

C -S HELLO.C (или C -S -O HELLO.C, если необходим оптимизатор глазка)

Это создает файл HELLO.AS, который может быть обработан ассемблером HI-TECH ZAS. Я сгенерировал утилиту:

AS2MAC HELLO.AS >HELLO.MAC

AS2MAC преобразует файл AS, созданный C, в формат MAC, подходящий для Microsoft M80. ПРИМЕЧАНИЕ. Символ '>' не является обязательным. AS2MAC всегда записывает вывод в консоль, но поддерживается перенаправление. AS2MAC.C включен. Внутри LIBCREL.LBR есть версия - игнорируйте ее (она устарела).

Файлы AS, созданные HI-TECH C, имеют прямые ссылки (jp 1f ... 1:), которые не поддерживает макроассемблер M80. Также psect, .high. &. Если используется опция -O (оптимизировать), генерируются символы, начинающиеся с L, и конфликтуют с символами l в нижнем регистре (M80 не чувствителен к регистру). Константы с плавающей точкой (deff) заменяются четырехбайтовыми последовательностями. Ключевое слово global конвертируется в extern при необходимости. Расширенные коды условий заменяются (например, jp llt). Чтобы выполнить преобразование, AS2MAC выполняет два прохода по файлу AS. Он всегда записывает окончательный MAC в стандартный вывод. AS2MAC **не** является общей утилитой - он **только** преобразует выходные данные из C. После преобразования:

M80 =HELLO

и компоновки с помощью

L80 ZCRTCPM,HELLO,HELLO/N/E

Я преобразовал библиотеку C в LIBC.REL, а библиотеку с плавающей точкой в LIBF.REL. Обратите внимание, что ZCRTCPM имеет .REQUIRE LIBC. Если используется с плавающей запятой, к вызову редактора связей необходимо добавить LIBF/S.

Поскольку компоновщик L80 понимает только 6 значащих символов, изменяются следующие имена (и STRING.H исправлено):

strncpy -> stncpy

strncat -> stncat

strncmp -> stncmp

Также изменены некоторые внутренние имена - предоставлен обновленный исходный код библиотеки.

UEXIT.MAC был изменен, чтобы записать код возврата в местоположение 0x7e, а не 0x80, как это делал оригинальный код Hi-Tech. 0x80 уничтожается CCP перед выполнением следующей программы. 0x7e безопаснее.

Я также исправил драйвер (C.COM) для ссылки на LINQ.COM и CREQ.COM, чтобы избежать проблем при объединении с другими цепями инструментов. LINK.COM теперь называется LINQ.COM, а CREF.COM называется CREQ.COM.

Aztec C - это следующий наиболее полезный компилятор C для моих нужд. Но Aztec не предлагает прототипов функций. Компилятор Whitesmith C - это то, что я использовал "в прежние времена", но он тоже не ANSI (нет прототипов), имеет нестандартную библиотеку, очень обособленную цепочку инструментов (ассемблер - это нечто под названием "A-natural"). Компилятор Whitesmith также требует большой объем памяти (62KB TPA минимально) и генерирует код только для 8080. Компилятор Hi-Tech работает с CP/NET 1.2, которая является моей предпочтительной средой. Компилятор Aztec генерирует код как 8080, так и для Z80, Hi-Tech C - только для Z80. Однако я могу принять это ограничение.

По этим причинам я выбрал Hi-Tech в качестве своего предпочтительного компилятора языка Си. Однако я не хочу покидать MACRO-80, так как это все еще предпочтительная ассемблерная система на CP/M. Таким образом родился проект MTREL.

Для обновления модуля в библиотеке с lib80 используйте следующий подход. Например, допустим, вы хотите обновить UEXIT.MAC. Извлеките его из LIBCREL.LBR и скомпилируйте, затем замените в LIBC.REL:

M80 =UEXIT

LIB80 NEW=LIBC<..UEXIT-1>,UEXIT,LIBC<UEXIT+1..>

PIP LIBC.REL=NEW.REL

Идея состоит в том, чтобы поддерживать тот же порядок, что и в LIBC.REL. Возможно, есть более простой подход, но я всегда так делал.

Пример предоставляется:

C -S SAMPLE.C

AS2MAC SAMPLE.AS >SAMPLE.MAC

M80 =SAMPLE

L80 ZCRTCPM,SAMPLE,SAMPLE/N/E

**ZSID**

**Отладчик Z-80 для CP/M**

Алан Р. Миллер {Alan R. Miller}

Interface AGE август 1980

Одной из стандартных программ, поставляемых с CP/M, является DDT (Dynamic Debugging Tool), независимая программа, используемая для тестирования и отладки других программ.

В сравнении с DDT, SID (Symbolic Instruction Debugger) является более продвинутым отладчиком. Обычная версия работает как на 8080, так и на Z-80, но символические функции относятся только к стандартной мнемонике языка ассемблера Intel 8080. ZSID - это версия SID для Z-80, использующая официальную мнемонику Zilog. Поскольку ZSID написан с инструкциями Z-80, он не будет работать на машине 8080.

Программы CP/M обычно выполняются в области переходных программ (TPA), начиная с 100 hex. Когда SID выполняется, он первоначально загружается в TPA, а затем автоматически перемещается в верхнюю часть используемой памяти CP/M. Это оставляет начало TPA доступным для тестирования других программ. После установки SID его можно направить на загрузку отдельной программы в TPA. Кроме того, отдельная программа может быть загружена вместе с SID путем включения имени программы в командную строку. Например, утверждение:

**B>A:SID PAYROLL.COM**

сначала загрузит SID, затем SID загрузит программу payroll.com. Если программа представляет собой шестнадцатеричный файл, то SID декодирует его, размещая в требуемом месте.

**Возможности монитора**

SID содержит большинство обычных функций монитора. Например, команда: 'D100, 17f' будет отображать ячейки памяти от 100 до 17f hex. Каждое местоположение будет указано в шестнадцатеричном формате, а также в ASCII, если символы доступны для печати. DDT и версия 8080 SID выводят значения ASCII в конце той же строки, что и шестнадцатеричный код. Однако ZSID помещает символы ASCII в следующую строку под соответствующим шестнадцатеричным кодом. Этот формат более удобен для узких видеоэкранов.

Блок памяти можно переместить с помощью команды **M**. Команда **S** позволяет пользователю изменять (устанавливать) места в памяти. Шестнадцатеричное сложение и вычитание можно выполнить с помощью команды **H**. Команда **X** позволяет пользователю исследовать и изменять регистры процессора. К ним относятся регистры общего назначения, набор альтернативных регистров, регистр флагов, счетчик программ, указатель стека и регистры индексов X и Y. В отладчике отсутствует команда поиска, а также команда для ввода или вывода данных через порт ввода-вывода.

Команда **L** производит символическую разборку машинного кода с использованием официальной мнемотехники Zilog. Команда **А** позволяет непосредственно кодировать мнемонику на ассемблере. Ассемблеры Z-80 не являются последовательными в своих мнемонических наборах. Рассмотрим инструкции по вычитанию:

**SBC А, С**

**SBB А, С**

Первая - это официальная мнемоника Zilog для 8-битного вычитания с заимствованием. Вторая также разрешена некоторыми ассемблерами, так как она напоминает мнемонику 8080. Но ZSID не примет вторую версию во время операции **A**.

Еще одна особенность Z-80 возникает с набором мнемоник сложения и вычитания. Zilog использует два операнда для операций сложения и вычитания в 16-битной и 8-битной версиях. Тем не менее, только один операнд используется для 8-битной инструкции 'sub', поскольку нет соответствующей 16-битной операции.

16-bit 8-bit

**ADC HL.DE ADC A,E**

**ADD HL.DE ADD A,E**

**SBB HL,DE SBB A,E**

**SUB E**

Поскольку аккумулятор всегда является регистром назначения для 8-битных операций, аргумент на самом деле не нужен. По этой причине некоторые ассемблеры не требуют операнда назначения ни для одной из 8-битных операций сложения и вычитания. ZSID, однако, требует двух операндов для всех команд 'adc', 'add' и 'sbb'. Другой распространенный вариант допускает два операнда для команды 'sub':

**SUB A, E**

делает инструкцию похожей на другие операции сложения и вычитания. Эта форма была включена в ZSID, хотя она не является мнемоникой Zilog.

**Выполнение программы с помощью SIO**

Несколько команд позволяют пользователю выполнять программу под управлением SID. Предположим, что программа работает должным образом, пока не достигнет адреса 244. Программа может быть загружена с помощью SID, затем может быть дана команда 'G100,244'. Это запустит программу по адресу 100 hex, первый аргумент, и установит точку останова по адресу 244, второй аргумент. Если программа попадет по адресу 244, управление автоматически вернется к SID. Это происходит потому, что SID заменяет исходный байт по адресу 244 инструкцией RST 38H. Когда управление возвращается к SID, исходный байт восстанавливается. Затем пользователь может проверить регистры процессора, изменить их при желании и продолжить выполнение программы с помощью другой команды **G**.

Команда **P** аналогична команде **G**. Она позволяет команде пройти определенное количество раз, прежде чем программа вернет управление в SID. Эта команда удобна для отладки циклов. Команда **T** может использоваться для пошагового прохождения программы. Эту команду также можно использовать для отслеживания предыдущих шагов.

**Аргументы к командам SID**

Несколько различных типов аргументов могут быть заданы с помощью команд SID. Основное значение по умолчанию - шестнадцатеричное, но это можно изменить с помощью префикса. Знак # предшествует десятичному числу, а перед символом ASCII используется апостроф. Шестнадцатеричные аргументы ZSID немного отличаются от аргументов версии 8080 SID. Последний принимает любое действительное шестнадцатеричное число. Однако в случае ZSID первая цифра шестнадцатеричного числа должна быть одной из десятичных цифр 0-9, т. е. Должен использоваться предшествующий ноль, если первая цифра A-F. Это ограничение необходимо, потому что мнемоника Z-80 имеет структуру, отличную от мнемоники 8080. Рассмотрим инструкции 8080:

**MOV A,C**

**MVI A,C**

Первая - это перемещение между регистрами, вторая инструкция загружает значение C hex в аккумулятор. Версия SID 8080 может различать две инструкции, поскольку мнемоника различна. Однако эквивалент Z-80 использует одну и ту же мнемонику для обеих команд:

**LD A,C**

**LD A,0C**

и поэтому должна использоваться предшествующий ноль, чтобы отличить шестнадцатеричное значение C от имени регистра C. Аргументы могут быть выражены относительно предыдущего адреса. Например, команда 'D100, +5' отобразит память от адреса 100 до 105.

**Символические ссылки**

Ассемблеры Digital Research MAC и ASM и ассемблер Microsoft Macro-80 с его загрузчиком связей могут создавать отдельную таблицу символов. SID может быть направлен на загрузку этой таблицы символов вместе с тестируемой программой. Символьные ссылки из таблицы символов затем можно использовать в качестве аргументов для SID. Символы можно использовать тремя способами.

Если ему предшествует десятичная точка, символ используется в качестве указателя. Предположим, что таблица символов содержит метку 'outhz', соответствующую адресу программы 163. Затем команда 'l.outhx, + #20' разберет 21 (десятичное) положение, начиная с адреса 'outhx'. Кроме того, записи таблицы символов будут по возможности указываться в листинге как в качестве адресов, так и в качестве операндов в инструкциях. Типичная разборка, если загружена таблица символов, может выглядеть следующим образом:

**OUTHX:**

**163 CALL 0167 .HEX**

**166 LD A,B**

**HEX:**

**167 AND 0F**

Значение, сохраненное по адресу, соответствующему символу, также может быть получено. Если символу предшествует знак @, используется 16-битное значение в указанном местоположении. С другой стороны, если символу предшествует знак равенства, выбирается значение байта в этом месте. Предположим, что символ 'iobyte' был определен как местоположение 3. Затем текущее значение, сохраненное по этому адресу, можно получить с помощью команды:

**H = IOBYTE**

**Программы с аргументами**

Некоторые программы CP/M требуют один или два аргумента в командной строке. Утверждение:

**A>B:LIST B:SORT.PAS**

даст команду CP/M загрузить и выполнить файл 'list.com' с диска B. Кроме того, имя файла 'b:sort.pas' помещается в память в блок управления файлами, начиная с адреса 5C hex. Наконец, CP/M запускает программу 'list', переходя по адресу 100 hex. Такие программы, как list, требующие аргументы в командной строке, можно отлаживать с помощью SID. В этом случае первой командой будет:

**A>SID LIST.COM LIST.SYM**

Эта строка загружает SID, list и таблицу символов list. Следующим шагом является настройка блока управления файлами. Команда:

**#IB:SORT.PAS**

запишет запрошенное имя файла в блок управления файлами. Теперь программу list можно запустить под управлением SID. Команда **G**, **P** или **T** задается с соответствующей точкой останова, поэтому в какой-то момент управление вернется к SID.

Когда программа запускается под управлением SID, стек изначально помещается по адресу 100 hex. Тестируемая программа должна изменить указатель стека на что-то другое, если необходимо выполнить чтение с диска. Этот шаг необходим, поскольку CP/M использует область с 80 до FF hex в качестве буфера для дисковых операций.

**Недокументированные инструкции Z-80**

Покупателям ZSID предоставляется обычное руководство пользователя SID и удобное дополнение, которое обобщает официальный набор инструкций Zilog. Более полное изложение инструкций доступно от Zilog. Но есть много инструкций Z-80, которые не были задокументированы Zilog. Большинство из них являются 8-битными операциями, включающими один из двух 16-битных регистров индекса. Все официальные инструкции включают 16-битные операции для регистров индекса.

Предположим, что мы хотим переместить регистр C в младший байт регистра IX. Один из способов сделать это, используя официальные инструкции:

**PUSH BC**

**POP IX**

Но этот метод также переместит регистр B в старший байт IX. Более прямой способ выполнить перемещение - использовать одну из недокументированных инструкций. Мнемоника языка ассемблера, которая может генерировать необходимый код:

**DEFB 0DDH**

**LD L,С**

Байт DD предшествует всем недокументированным 8-битным операциям IX, а FD предшествует всем недокументированным 8-битным операциям IY. Затем появление L или H в оставшейся части инструкции интерпретируется процессором Z-80 как байт младшего или старшего разряда соответствующего индексного регистра.

Переход из регистра C в IX можно выполнить с помощью ZSID. Сначала запишите значение DD в память командой установки:

**#S4000**

**4000 00 DD**

**4001 00 .**

Затем используйте команду **A** для ввода операции перемещения:

**#A4001**

**4001 LD L,C**

**4002**

Загрузите регистр C со значением 12 с помощью команды **X**:

**#XB**

**BC 0000 12**

Затем отобразите все регистры процессора с помощью команды **X**. Запустите эту короткую программу с помощью команды **G**, убедившись, что вы установили точку останова, чтобы управление вернулось к SID:

**#G4000, 4002**

Наконец, проверьте регистры процессора с помощью команды **X**. Младшие 8 битов регистра IX теперь будут содержать значение 12.

Все обычные 8-битные инструкции Z-80, включающие регистры H и L, могут использоваться таким образом. Если обычной инструкции предшествует DD, то операция выполняется с соответствующей половиной регистра IX. Если байт FD, то используется регистр IY. Операции включают в себя перемещение между регистрами:

**LD H,r**

**LD r,H**

**LD L,r**

**LD r,L**

где r - один из регистров общего назначения A, B, C, D, E. Регистры H и L теперь относятся к индексным регистрам. Таким образом, инструкция:

**DEFB 0DDH**

**LD H,L**

переместит нижнюю половину IX в верхнюю половину IX. Также могут быть выполнены непосредственные инструкции. Нижняя половина IY может быть загружена значением 7, используя инструкции:

**DEFB 0FDH**

**LD L,7**

Подобным образом могут использоваться также 8-разрядные арифметические и логические операции, включая команды сдвига и поворота.

Эти недокументированные инструкции Z-80 были включены в программное обеспечение PDS (Program Development System) Аллена Эшли {Allen Ashley}. Но есть несколько потенциальных проблем с использованием недокументированных кодов. Непосредственная проблема заключается в том, что ZSID не будет правильно декодировать инструкции. Кроме того, эти инструкции могут быть недоступны на будущих версиях Z-80, особенно если чип получен от поставщика второго источника. Тем не менее, эти 8-битные инструкции дают программисту некоторые дополнительные регистры, которые иногда могут понадобиться.

**Программа Despool**

Отдельная программа Despool доступна от Digital Research. Эта программа позволяет использовать принтер, пока пользователь выполняет другие действия на видеотерминале. Поскольку процессор работает намного быстрее, чем периферия, все периферийные устройства могут работать одновременно. Большие компьютеры с основной структурой и миникомпьютеры обычно работают именно подобным образом. В микрокомпьютерах одновременно работает только одно периферийное устройство.

Диспетчер очереди KLH (IA, апрель 79) управляет работой принтера различными способами. Им можно управлять из Basic, из редактора или с системного уровня. Он может работать только с 8-дюймовыми дисками с мягким сектором. Despool может работать с любой системой CP/M и любым типом диска. К сожалению, файлы могут быть записаны в буферный файл только с системного уровня, и принтер на мгновение останавливается каждый раз, когда используется консольная клавиатура. Тем не менее, Despool Digital Research лучше, чем ничего.

**The CP/M® Z-8O® Microcomputer**

**ZSID™**

**SYMBOLIC INSTRUCTION DEBUGGER**

**Перечень команд**

**Версия Z-80**

**DIGITAL RESEARCH**

**Авторское право**

Авторское право © 1979, 1981, Digital Research. Все права защищены. Никакая часть этой публикации не может быть воспроизведена, передана, расшифрована, сохранена в поисковой системе или переведена на любой язык или компьютерный язык, в любой форме или любыми средствами, электронными, механическими, магнитными, оптическими, химическими, ручными или другими, без предварительного письменного разрешения Digital Research, Почтовый ящик 579, Пасифик-Гроув, Калифорния, 93950.

**Товарные знаки**

CP/M является зарегистрированным товарным знаком Digital Research. Z-80 является зарегистрированным товарным знаком Zilog. ZSID является торговой маркой Digital Research.

Вся информация, содержащаяся в настоящем документе, является собственностью Digital Research.

**Об этом руководстве**

Отправной точкой для этого PDF-файла был обычный текст, содержащийся в файле **zsid.txt**[[1]](#footnote-1) (Отсканированный из руководства CP/M), заархивированном в файле **zsid-m.zip**, который был загружен с [www.cpm.z80.de](http://www.cpm.z80.de), неофициальный веб-сайт CP/M.

Пожалуйста, войдите на этот сайт, чтобы узнать, как вы можете помочь сообществу CP/M, работающему над руководствами CP/M.

Мигель И. Гарсия Лопес, 24 Мая 2007.

**Перечень команд ZSID**

**1.1 Запуск**

|  |  |
| --- | --- |
| **ZSID** | Запускает ZSID без тестовой программы. |
| **ZSID x.y** | Загружает тестовую программу **x.y** (**y** обычно COM). |
| **ZSID x.HEX** | Загружает **x.HEX** в формате Intel "hex". |
| **ZSID x.UTL** | Загружает и выполняет утилиту **x**. |
| **ZSID x.y u.v** | Загружает **x.y** с таблицей символов **u.v** (обычно x.SYM). |

Пример:

**ZSID SORT.COM SORT.SYM**

**1.2 Ответ**

|  |  |
| --- | --- |
| **#** | Указывает о готовности ZSID принимать команды. |
| **SYMBOLS** | Указывает, что машинный код загружен, начинается загрузка таблицы символов. |
| **NEXT PC END**  **nnnn pppp eeee** | Показывает успешный машинный код и/или загрузку символов, где **nnnn**, **pppp** и **eeee** - шестнадцатеричные значения, дающие следующее незаполненное местоположение кода машины, начальный счетчик программы (далее по тексту **PC**)[[2]](#footnote-2) и последнее свободное место в памяти, соответственно. |

**1.3 Буквенные команды**

**2.5 Переход к программе**

|  |  |
| --- | --- |
| **G** | Запускает тестируемую программу с текущего PC без точек останова. Исполнение в реальном времени. |
| **Gp** | Эквивалентна предыдущей, но устанавливает PC=**p** перед выполнением, |
| **G,a** | Запускается с текущего PC с точкой останова в местоположении **a**, |
| **Gp,a** | Аналогична предыдущей, но устанавливает PC в **p**. |
| **G,a,b** | Эквивалентна третьей, но устанавливает точки останова на **a** и **b**, |
| **Gp,a,b** | в то время как эта перед выполнением устанавливает PC равным **p**. |
| **-G**... | Аналогична предыдущим, за исключением того, что «точки прохода» не отслеживаются до тех пор, пока соответствующий счетчик проходов не станет равным нулю (см. Команду **P**). |

При обнаружении точки останова (или внешнего RST 7) адрес останова печатается в виде:

**\*nnnn**

и необязательные точки останова очищаются.

Символ "**^**" в выражении создает самое верхнее значение стека, которое используется для установки разрыва после вызова подпрограммы. Поскольку в подпрограмме возникла точка останова, команда

**G,^**

продолжает выполнение с возвращаемым набором точек останова.

Примеры:

G100

G100,103

G.CRLF,.PRINT,#1024

G@JMPVEC+=I,.ENDC,.ERRC

G,.errsub

G,.ERRSUB,+30

-G100,+10,+10

**2.7 Входная строка**

**Ic1c2...cn**

Инициализирует стандартные области низкой памяти для команды **R** или тестируемой программы, как если бы символы **c1** - **cn** были прочитаны и настроены на уровне командного процессора консоли. FCB по умолчанию инициализируются, и буфер по умолчанию устанавливается в исходную строку ввода.

Примеры:

I x.dat

ix.inp y.out

I a:x.inp b:y.out $-p

ITEST.COM

I TEST.HEX TEST.SYM

**2.10 Счетчик проходов**

|  |  |
| --- | --- |
| **Pp** | Устанавливает точку прохода по адресу **p** с количеством проходов 1. |
| **Pp,c** | Устанавливает точку прохода p с количеством проходов **c**. |
| **P** | Отображает активные точки прохода и счетчики. |
| **-Pp** | Очищает точку прохода в точке p (эквивалентно **Pp,0**). |
| **-P** | Очищает все точки прохода. |

"Точка прохода" - это местоположение счетчика программ, которое необходимо отслеживать во время выполнения тестируемой программы. Точка прохода имеет связанный "счетчик проходов" в диапазоне 1-FF (0 - #255), который уменьшается каждый раз, когда тестируемая программа выполняет адрес точки прохода. Когда счетчик проходов достигает 1, точка прохода становится постоянной точкой останова, а счетчик проходов остается на 1. В отличие от временной точки останова (см. **G**), точки прохода с счетом проходов равным 1 останавливаются после выполнения инструкции по адресу прерывания. До 8 точек прохода могут быть активны в любое время. Регистры ЦП отображаются при выполнении точки прохождения с заголовком

**nn PASS hhhh .ssss**

отображая счетчик проходов **nn** и адрес **hhhh** с необязательным символом **ssss**.

Во время выполнения команд **-G** или **-U** регистры не отображаются пока счетчик проходов не станет равен 1. Выполнение может быть прервано во время прохода с помощью кнопка клавиша стирания.

Примеры:

P100,ff

P.BDOS

P@ICALL+30,#20

-P .CRLF

**2.11 Считывание кода / символов**

Команда **I** настраивает файлы кода и символов для последующей загрузки с помощью команды **R**.

|  |  |
| --- | --- |
| **R** | (1) Считывает необязательный код и необязательные символы при подготовке к тестированию программы, |
| **Rd** | (2) аналогична, но загружает код и/или символы со значением смещения. |

Последовательность:

**I x.y R**

настраивает файл машинного кода **x.y** (**y** обычно является COM) и считывает машинный код в область переходных программ. Если **y** - шестнадцатеричный, то файл должен быть в формате Intel "hex". Последовательность:

**I x.y u.v R**

также читается файл символов **u.v** (**u** обычно совпадает с **x**, а **v** обычно является SYM). Форма:

**I \* u.v R**

пропускает загрузку машинного кода и читает только файл символов.

Когда указан файл символов, ответ

**SYMBOLS**

показывает начало операции чтения файла символов. Таким образом, ошибка "**?**" перед сообщением SYMBOL указывает на ошибку чтения машинного кода, в то время как "**?**" после сообщения SYMBOL показывает ошибку чтения файла символов.

Примеры:

**I COPY.COM**

**R**

**I SORT.HEX SORT.SYM**

**R**

**I merge.com merge.sym**

**R1OOO**

**I \* test.sym**

**R-#256**

**2.13 Режим трассировки**

|  |  |
| --- | --- |
| **Tn** | (1) отслеживает **n** шагов программы, показывая состояние ЦП на каждом шаге, в то время как |
| **T** | (2) отслеживает один шаг. |
| **Tn,c** | (3) используется с утилитами ZSID и "вызывает" функцию утилиты **c** на каждом шаге трассировки. |
| **T,c** | (4) подобна (3), но отслеживает только один шаг. |
| **-T ...** | (5) аналогична (1) - (4), но отключает символы. |
| **TW ...** | (6) аналогична (1) - (4), но выполняет "трассировку без вызова", показывая только локальное исполнение. |
| **-TW ...** | (7) аналогична форме (6) с отключенными символами. |

Примеры:

**T100**

**T#30,.COLLECT**

**-TW=I,3E03**

**# iz-cpm -- среда CP/M 2.2**

**## Что это?**

Это среда выполнения CP/M 2.2. Она предоставляет все необходимое для запуска стандартного CP/M бинарного файла Z80.

Использует мою [iz80](<https://github.com/ivanizag/iz80>) библиотека для эмуляции Z80.

Написана на языке Rust

**## Установка**

Извлеките [последний zip](<https://github.com/ivanizag/iz-cpm/releases>) для Linux, MacOS или Windows. По желанию запуск "download.sh" или "download.bat" для загрузки системного диска CP/M 2.2, Microsoft Basic, Turbo Pascal, Lisp и некоторых игр.

**## Примеры использования**

Выполните "iz-cpm", чтобы открыть командную строку CP/M (CCP) в текущем каталоге:

casa@servidor:**~/software/cpm22$ ls**

ASM.COM CCSINIT.COM DISKDEF.LIB iz-cpm STAT.COM

CBIOS.ASM CCSYSGEN.COM DUMP.ASM LOAD.COM STDBIOS.ASM

CCBIOS.ASM CPM24CCS.COM DUMP.COM MOVCPM.COM SUBMIT.COM

CCBOOT.ASM DDT.COM ED.COM PIP.COM SYSGEN.COM

-CCSCPM.251 DEBLOCK.ASM GENMOD.COM RLOCBIOS.COM

casa@servidor:**~/software/cpm22$ ../../iz-cpm**

iz-cpm https://github.com/ivanizag/iz-cpm

CP/M 2.2 Copyright (c) 1979 by Digital Research

Press ctrl-c ctrl-c to return to host

A>**dir**

A: CCSINIT COM : MOVCPM COM : CPM24CCS COM : STAT COM

A: SYSGEN COM : STDBIOS ASM : IZ-CPM : LOAD COM

A: DISKDEF LIB : ASM COM : RLOCBIOS COM : DUMP COM

A: -CCSCPM 251 : CCBIOS ASM : DUMP ASM : CCSYSGEN COM

A: DEBLOCK ASM : ED COM : SUBMIT COM : DDT COM

A: PIP COM : CBIOS ASM : CCBOOT ASM : GENMOD COM

A>

Выполните "iz-cpm" с файлом для непосредственного выполнения двоичного файла CP/M, минуя CPP:

casa@servidor:**~/code/z80/iz-cpm$ ./iz-cpm software/OBASIC.COM**

44531 Bytes free

BASIC Rev. 4.51

[CP/M Version]

Copyright 1977 (C) by Microsoft

Ok

print "hello"

hello

Ok

Подключите до 16 каталогов в качестве дисков CP/M:

casa@servidor:~$ **./iz-cpm --disk-a software/cpm22 --disk-b software/zork --disk-e .**

iz-cpm https://github.com/ivanizag/iz-cpm

CP/M 2.2 Copyright (c) 1979 by Digital Research

Press ctrl-c ctrl-c to return to host

A>b:

B>dir

B: ZORK2 DAT : ZORK1 COM : ZORK3 DAT : ZORK3 COM

B: FILE\_ID DIZ : ZORK1 DAT : ZORK1 SAV : ZORK2 COM

B>zork1

ZORK I: The Great Underground Empire

Copyright (c) 1981, 1982, 1983 Infocom, Inc. All rights

reserved.

ZORK is a registered trademark of Infocom, Inc.

Revision 88 / Serial number 840726

West of House

You are standing in an open field west of a white house, with

a boarded front door.

There is a small mailbox here.

>

## **Использование**

iz-cpm https://github.com/ivanizag/iz-cpm

CP/M 2.2 Copyright (c) 1979 by Digital Research

Press ctrl-c ctrl-c to return to host

ИСПОЛЬЗУЙТЕ:

iz-cpm [ФЛАГИ] [ОПЦИИ] [АРГУМЕНТЫ]

ФЛАГИ:

|  |  |
| --- | --- |
| -t, --call-trace | Отслеживание вызовов BDOS, исключая ввод-вывод на экран |
| -T, --call-trace-all | Отслеживает вызовы BDOS и BIOS |
| -z, --cpu-trace | Отслеживает выполнение инструкций Z80 |
| -h, --help | Печатает справочную информацию |
| -s, --slow | Работает медленнее |
| -V, --version | Печатает информацию о версии |

ОПЦИИ:

|  |  |
| --- | --- |
| -a, --disk-a <путь> | каталог для сопоставления диска A: [по умолчанию: .] |
| -b, -- disk-b <путь> | каталог для сопоставления диска B: |
| -c, -- disk-c <путь> | каталог для сопоставления диска C: |
| -d, -- disk-d <путь> | каталог для сопоставления диска D: |
| -- disk-e <путь> | каталог для сопоставления диска E: |
| -- disk-f <путь> | каталог для сопоставления диска F: |
| -- disk-g <путь> | каталог для сопоставления диска G: |
| -- disk-h <путь> | каталог для сопоставления диска H: |
| -- disk-i <путь> | каталог для сопоставления диска I: |
| -- disk-j <путь> | каталог для сопоставления диска J: |
| -- disk-k <путь> | каталог для сопоставления диска K: |
| -- disk-l <путь> | каталог для сопоставления диска L: |
| -- disk-m <путь> | каталог для сопоставления диска M: |
| -- disk-n <путь> | каталог для сопоставления диска N: |
| -- disk-o <путь> | каталог для сопоставления диска O: |
| -- disk-p <путь> | каталог для сопоставления диска P: |

АРГУМЕНТЫ:

|  |  |
| --- | --- |
| <CMD> | образ z80 для запуска |
| <ARGS> | параметры для данной команды |

**## Функции**

* Выполнение бинарных файлов 8080 и Z80 поверх CP/M
* Прямое использование файловой системы хост-компьютера.
* Эмуляция терминала ADM-3A, используемая в компьютерах KAYPRO
* Эмуляция Z80 проверена ZEXALL
* Отслеживание исполнения Z80
* Трассировка BDOS и BIOS
* Переносимость, работает в Linux, MacOS и Windows.

**## Как это работает**

[CP/M] (<https://en.wikipedia.org/wiki/CP/M> ) был разработан для переноса на различные устройства с использованием процессоров Intel 8080, Intel 8085 или Zilog Z80 благодаря многоуровневой архитектуре. Основными частями CP/M являются:

* BIOS: базовая система ввода / вывода. Интерфейс с оборудованием. CP/M определяет очень маленький интерфейс: 16 точек входа для управления вводом / выводом и доступом к секторам диска. Поставщики оборудования предоставили BIOS для своего устройства на основе исходного кода, доступного в дистрибутиве CP/M.
* BDOS: базовая дисковая операционная система. Основной компонент CP/M. Предоставлено Digital Research и является общим для всех систем. Предоставляет точки входа высокого уровня, которые будут использоваться разработчиками приложений.
* CCP: консольный командный процессор. Командная строка CP/M.

Чтобы эмулировать эту среду с помощью файловой системы хоста, мы должны предоставить замену BDOS-трансляции, поскольку мы не хотим ретранслировать абстракции секторов физического диска в BIOS. Основными компонентами являются:

* эмулятор Z80. Он использует [iz80](<https://github.com/ivanizag/iz80>)
* эмулятор [BIOS](src/bios.rs). Только точки входа ввода-вывода. Теоретически, в этом нет необходимости, но некоторые программы используют его напрямую, минуя BDOS.
* эмулятор [BDOS](src/bdos.rs). Перехватывает вызовы и выполняет код на хосте.
* CPP. Эмуляция CPP отсутствует. Используется двоичный файл CPP из CP/M 2.2..
* эмулятор [Terminal](src/terminal.rs). CP/M не определяет, как должен работать терминал. Приложения должны быть осведомлены и обычно могут быть настроены для нескольких ведущих опций, таких как ADM-3a, VT-52, Hazeltine 1500 и Osborne. Этот эмулятор поддерживает ADM-3a, используемый также на очень популярных компьютерах Kaypro.

**## Полезные ссылки**

- [Неофициальный Web сайт](<http://www.cpm.z80.de/>)

- [Руководства операционной системы] (<http://www.gaby.de/cpm/manuals/archive/cpm22htm/>)

- [Программы CP/M в ретро архиве retroarchive.org](<http://www.retroarchive.org/cpm/>)

## Необходимо сделать

- Надлежащие документы

- Опция только для чтения на уровне файла (я не буду этого делать, хост может это контролировать)

- Поддержка BIOS для перфокарт (нет)

- Поддержка BIOS для доступа к дискам на уровне дорожка/сектор (не требуется)

1. В этом документе для улучшения восприятия информации описание команд совмещено с формами их написания. [↑](#footnote-ref-1)
2. Добавлено при переводе [↑](#footnote-ref-2)