Санкт-Петербургский политехнический университет Петра Великого
Кафедра компьютерных систем и программных технологий

Техническое задание

Курсовая работа

Дисциплина: Низкоуровневое программирование

Тема: Симулятор 8-битного процессора intel 8051

Студент группы 23531/2
_____ Н.С. Макаревич

Преподаватель
_____ М.Х. Ахин
«___» _____ 2018 г.

Санкт-Петербург, 2018 г.

Содержание

1 Введение

1.1 Назначение

Данный программный продукт может быть использован в качестве исполнителя программ, отладочного средства для программ под intel 8051, а также для демонстрации работы программ. В связи с тем, что микропроцессор 8051 был создан 1980 году, на данный момент трудно найти реальный работающий экземпляр, причём часто их цена сильно завышена.
Если требуется написать программу для этого процессора, то для её отладки лучше использовать симулятор. Он бесплатен, а также позволяет отслеживать порядок инструкций и состояние памяти.

1.2 Краткий обзор

Данная программа должна симулировать работу процессора intel 8051, а именно работу с памятью и исполнение инструкций. Физические интерфейсы микроконтроллера программа не отражает.
Пользователь составляет файл,описывающий состояние памяти устройства в начальный момент времени. Файл может быть либо бинарным (код программы), либо написанным в специальном текстовом формате, который описан ниже. В текстовом формате также можно использовать дополнительные ключевые слова для отладки.
Симулятор должен исполнять программы в соответствии со спецификацией процессора 8051, то есть выполнять над виртуальной памятью и регистрами те же действия, которые заявлены у реального процессора, и в такой же последовательности.
Скорость выполнения программы может отличаться от скорости реального процессора в большую или меньшую сторону, на это ограничений не накладывается. При желании скорость обработки инструкций можно замедлить при помощи аргументов командной строки.
Также данный симулятор должен поддерживать возможность передать необходимые параметры (не для отладки) только через аргументы командной строки, чтобы облегчить работу с симулятором других программ и сценариев.

2 Описание

2.1 Система команд процессора

Набор инструкций, которые может обрабатывать симулятор, должен соответсововать тому набору инструкций, которыми оперирует реальный процессор intel 8051.
Описание всех инструкций, регисторв и поведения процессора приведено в книге

Горюнов А.Г., Ливенцов С.Н. Архитектура микроконтроллера Intel 8051
Томск: Изд-во ТПУ, 2005. - 86 с.

2.2 Взаимодействие с пользователем

Эта программа имеет консольный интерфейс и запускается из командной строки с необходимыми параметрами.

Параметры:

-h --help
Показать краткую справку по командам симулятора.

-d --debug
Включает отладочные средства (реакцию на брэйкпойнты, вывод в консоль). Если флаг не установлен, то симулятор исполняет программу, ничего не выводя в консоль.

-i --infile
Имя входного файла.
Пример: -i myfile

-o --outfle
Имя выходного файла.
Так будет называться файл образа памяти, полученный после завершения программы.
Имя файла по-умолчанию – "memory".
Пример: -o myfile

-c --clk
Время задержки между исполнением инструкций. Количество милисекунд, целое беззнаковое число. По умолчанию задержки нет.
Пример: -c 1000

-v --verbose
Verbose режим. Отображает в реальном времени последовательность машинных команд, которые исполняет процессор. По умолчанию выключен.

-m --mode
Тип принимаемого на вход файла (bin или text)
Пример: -m bin

-b --break
Добавление брэйкпойнтов
В этом параметре передаётся адрес в шестнадцатеричном виде. На этом адресе будет добавлен брэйкпойнт.
^ означает, что брэйкпойнт сработает перед исполнением соответствующей инструкции. _ означает, что он сработает после её исполнения.
Пример: -b ^2C

--nobreak Игнорирование брэйкпойнтов
С этим флагом программа не будет останавливаться на брэйкпойнтах, обозначенных пользователем в файле или в командной строке.

-s --save
Добавление сэйвпойнтов

В этом параметре передаётся адрес в шестнадцатеричном виде. На этом адресе будет сделан снимок состояния процессора.
^ означает, что снимок будет сделан перед исполнением соответствующей инструкции. _ означает, что он будет сделан после её исполнения.
Пример: -s _2C

-z --convert
С этим флагом симулятор не исполняет программу, а просто преобразует входной бинарный файл в текстовый файл - образ памяти.

-e --end
Задание конечного адреса программы.
Адрес задаётся как шестнадцатеричное беззнаковое число.
Если счётчик PC превысит это значение, то программа будет остановлена, а конечное состояние сохранено в файл.
Пример: -e E6

--epm Флаг, включающий поддержку внешней памяти программ.

--edm Флаг, включающий поддержку внешней памяти данных.

--step Step-by-step режим. С этим флагом программа будет исполняться по одной инструкции по нажатию клавиши Enter.
Брэйкпойнты генерируются после каждой исполненной инструкции.

2.3 Текстовый формат ввода программы

Снимок состояния i8051
представляет собой JSON-структуру вида:

```
{
  "PC": "#.."
  "rga": "#..",
  "rgb": "#..",
  "rgc": "#..",
  ...
  "program": "...",
  "data": "...",

}
```

Состояние счётчика инструкций (PC), памяти программы (program) и памяти данных (data) задаётся в специальном текстовом формате.
Байты памяти идут по порядку. Пробелы, табуляция и переносы строки используются в качестве разделителя.

Числа и инструкции в памяти

Обычное число (8 двоичных разрядов) может быть записано в виде:

Шестнадцатеричного числа:	#FF
Двоичного числа:	11111111
Десятичного числа:	*255

Числа должны быть целыми и неотрицательными. Если симулятор прочтёт число больше 255, то программа завершится с ошибкой.

Также пользователь может писать названия инструкций и регистров латинскими буквами. Инструкции разделяются пробелами, символами табуляции или переносом строки. Регистр букв не учитывается.

Брэйкпойнты

В текстовое представление можно добавлять точки останова программы (breakpoint). Когда программа доходит до брэйкпойнта, то приостанавливается и пользователь может выполнять некоторые действия, прежде чем запустит ход выполнения программы дальше.
Пользователь может и сам остановить программу в нужном месте, нажав клавишу Enter. Тогда программа остановится после окончания исполнения текущей инструкции.
Можно просматривать значения, хранящиеся в определённых ячейках памяти программ и памяти данных. Для этого необходимо написать букву «p» или «d», которые указывают на место, откуда мы читаем значение. «p» - из памяти программы, «d» - из памяти данных.
Далее после буквы пользователь должен указать адрес ячейки в шестнадцатеричном виде и нажать клавишу Enter.
Например, чтобы прочитать из памяти данных значение, находящееся по адресу 88, достаточно написать d88 .
Также после остановки программы на брэйкпойнте пользователь может сохранить состояние процессора в файл, написав команду "save".
Команда "step"позволяет включать/выключать режим пошагового исполнения инструкций.
Брэйкпойнт записывается в формате ^BREAK или _BREAK
Символ ^ ставится перед BREAK, если программа должна быть приостановлена после предыдущей инструкции, а символ _ ставится, если программа должна быть приостановлена перед следующей инструкцией.

Точки сохранения снимка памяти

Можно добавлять в код точки сохранения (savepoint). Когда программа доходит до сейвпойнта, то дампит в бинарный файл состояние памяти и регистров в данный момент.
Сейвпойнт записывается в формате ^SAVE или _SAVE
Символы ^ и _ выполняют ту же функцию, что и в случае с брэйкпойнтами.

Брэйкпойнты и сейвпойнты не влияют на последовательность инструкций, которую в итоге исполняет процессор. Симулятор держит их в памяти отдельно и отслеживает, когда то или иное правило должно сработать.

Комментарии

Пользователь может включать в текст программы комментарии, которые будут проигнорированы симулятором, но сделают код более понятным. Они записываются в одинарных кавычках (' '). Комментарий может содержать любые символы кроме одинарных кавычек. Комментарии будут проигнорированы на этапе перевода текста в бинарный код и никак не повлияют на программу.