Практическое задание № 1. Разработка библиотеки mySimpleComputer. Оперативная память, регистр флагов, декодирование операций.

Цель работы

Изучить принципы работы оперативной памяти. Познакомиться с разрядными операциями языка Си. Разработать библиотеку mySimpleComputer, включающую функции по декодированию команд, управлению регистрами и взаимодействию с оперативной памятью.

Задание на лабораторную работу

1. Прочитайте главу 4 практикума по курсу «Организация ЭВМ и систем» (читать тут). Изучите принципы работы разрядных операций в языке Си: как можно изменить значение указанного разряда целой переменной или получить его значение. Вспомните, как сохранять информацию в файл и считывать её оттуда в бинарном виде.
2. Разработайте часть библиотеки mySimpleComputer, моделирующую работу оперативной памяти и прямого доступа к ней (используется для контроллера оперативной памяти центрального процессора и устройства ввода-вывода):
   - В качестве «оперативной памяти» используется массив целых чисел, определенный статически в рамках библиотеки (глобальная переменная внутри библиотеки). Размер массива равен 128 элементам. Тип элементов массива – целые числа (`int`). Массив, содержащий «оперативную память» доступен только функциям библиотеки (в пользовательском API он отсутствует);
   - `int sc_memoryInit (void)` – инициализирует оперативную память Simple Computer, задавая всем её ячейкам нулевые значения.
   - `int sc_memorySet (int address, int value)` – задает значение указанной ячейки памяти как value. Если адрес выходит за допустимые границы или value не соответствует допустимому диапазону значений, то функция возвращает `-1`, иначе завершается корректно и возвращает `0`;
   - `int sc_memoryGet (int address, int * value)` – возвращает значение указанной ячейки памяти в value. Если адрес выходит за допустимые границы или передан неверный указатель на value, то функция завершается со статусом `-1`. В случае успешного выполнения функции она завершается со статусом 0.
   - `int sc_memorySave (char * filename)` – сохраняет содержимое памяти в файл в бинарном виде (используя функцию write или fwrite). Если передан неверный указатель на имя файла или произошла какая-либо ошибка записи данных в файл, то функция завершается со статусом -1. В случае успеха функция завершается со статусом 0;
   - `int sc_memoryLoad (char * filename)` – загружает из указанного файла содержимое оперативной памяти (используя функцию read или fread). Если передан неверный указатель на имя файла или произошла какая-либо ошибка чтения данных из файла, то функция завершается со статусом -1, при этом содержимое «оперативной памяти» никак не изменяется (т.е. оно не должно портиться). В случае успеха функция завершается со статусом 0;
3. Разработайте часть библиотеки mySimpleComputer, моделирующую регистры Simple Computer:
   - регистры «Аккумулятор», «Счетчик команд», «Регистр флагов» - целые переменные (глобальные для библиотеки и недоступные напрямую пользователю);
   - `int sc_regInit (void)` – инициализирует регистр флагов значениями по умолчанию;

- `int sc_regSet (int register, int value)` – устанавливает значение указанного регистра флагов. Для номеров регистров флагов должны использоваться маски, задаваемые макросами (#define). Если указан недопустимый регистр, то функция завершается со статусом -1 и значение флага не меняется. Иначе статус завершения – 0. Флаг меняется в соответствии с правилами определения логического значения целой переменной, принятых в языке Си;
- `int sc_regGet (int register, int * value)` – возвращает значение указанного флага. Если указан недопустимый регистр и передан неверный указатель на значение, то функция завершается со статусом -1. Иначе статус завершения – 0.
- `int sc_accumulatorInit (void)` – инициализирует аккумулятор значением по умолчанию;
- `int sc_accumulatorSet (int value)` – устанавливает значение аккумулятора. Если указано недопустимое значение, то функция завершается со статусом -1 и значение аккумулятора не меняется. Иначе статус завершения – 0;
- `int sc_accumulatorGet (int * value)` – возвращает значение аккумулятора. Если передан неверный указатель на значение, то функция завершается со статусом -1. Иначе статус завершения – 0.
- `int sc_icounterInit (void)` – инициализирует счетчик команд;
- `int sc_icounterSet (int value)` – устанавливает значение счетчика команд. Если указано недопустимое значение, то функция завершается со статусом -1 и значение счетчика не меняется. Иначе статус завершения – 0;
- `int sc_icounterGet (int * value)` – возвращает значение счетчика команд. Если передан неверный указатель на значение, то функция завершается со статусом -1. Иначе статус завершения – 0.

4. Разработайте часть библиотеки mySimpleComputer, моделирующую часть устройства управления, отвечающую за кодирование и декодирование команды:
   - `int sc_commandEncode (int sign, int command, int operand, int * value)` – кодирует значение ячейки в соответствии с форматом команды Simple Computer и с использованием в качестве значений полей полученные знак, номер команды и операнд и помещает результат в `value`. Если указаны недопустимые значения для знака, команды или операнда, то функция завершается со статусом -1 и значение value не изменяется. В противном случае – статус завершения 0. Для знака, операнда и команды допустимыми являются все значения, которые соответствуют формату команды Simple Computer;
   - `int sc_commandDecode (int value, int * sign, int * command, int * operand)` – декодирует значение ячейки памяти как команду Simple Computer. Если декодирование невозможно, то функция завершается со статусом -1 и выходные параметры не меняю своего значения. Иначе статус завершения = 0;
   - `int sc_commandValidate (int command)` – проверяет значение поля «команда» на корректность. Если значение некорректное, то возвращается -1. Иначе возвращается 0;
5. Оформите разработанные функции как статическую библиотеку. Подготовьте заголовочный файл для неё. Доработайте систему сборки приложения таким образом, чтобы статическая библиотека mySimpleComputer собиралась при изменении любого из файлов с исходным кодом. Собранная библиотека должна располагаться в каталоге mySimpleComputer.

6. Разработайте часть устройства ввода-вывода (все данные, выводимые функциями в рамках данного практического задания, просто выводятся на экран, без формирования интерфейса консоли):
   - `void printCell (int address)` – выводит на экран содержимое ячейки оперативной памяти по указанному адресу. Формат вывода должен соответствовать заданию (ячейка выводится в декодированном виде);
   - `void printFlags (void)` – выводит значения флагов. Формат должен соответствовать заданию (выводятся либо _, либо буквы в заданной последовательности);
   - `void printDecodedCommand (int value)` – выводит переданное значение в десятичной системе счисления, в восьмеричной системе счисления, в шестнадцатиричной системе счисления и в двоичной системе счисления.
   - `void printAccumulator (void)` – выводит значение аккумулятора;
   - `void printCounters (void)` – выводит значение счетчика команд.
7. Разработайте тестовую программу pr01, которая должна использовать все созданные выше функции и библиотеку mySimpleComputer и выполнять следующие действия:
   - инициализировать оперативную память, аккумулятор, счетчик команд и регистр флагов;
   - установить произвольному количеству произвольных ячеек оперативной памяти произвольные значения. Вывести содержимое оперативной памяти (в декодированном формате по 10 ячеек в строке через пробел);
   - попробовать задать какой-нибудь ячейке оперативной памяти недопустимое значение и вывести статус завершения соответствующей функции;
   - установить произвольные значения флагов и вывести содержимое регистра флагов.
   - попробовать установить некорректное значение флага. Вывести статус завершения функции.
   - установить значение аккумулятора и вывести его на экран.
   - попробовать задать аккумулятору недопустимое значение и вывести статус завершения функции;
   - установить значение счетчика команд и вывести его на экран.
   - попробовать задать счетчику команд недопустимое значение и вывести статус завершения функции;
   - декодировать значение произвольной ячейки памяти и значение аккумулятора;
   - закодировать команду (любую допустимую из системы команд) и вывести полученное значение в разных системах счисления.
8. Итоговый исполняемый файл должен называться pr01 и располагаться в каталоге console.
9. Доработайте Makefile таким образом, чтобы автоматизированная сборка смогла собрать библиотеку и исполняемый файл pr01. Исполняемый файл должен собираться при изменении библиотеки или любого из исходных файлов каталога console. Также необходимо реализовать в Makefile искусственные цели по очистке каталога проекта от временных файлов (цель clean).

Контрольные вопросы

1. Что такое вентиль? Какие значения он может принимать?
2. Сколько вентилей необходимо, чтобы получить логические функции НЕ, ИЛИ-НЕ, И-НЕ, И, ИЛИ?
3. Что такое таблица истинности? Булева функция? Как они связаны между собой?
4. Как получить алгебраическую булеву функцию из таблицы истинности? И наоборот?
5. Каким образом можно синтезировать логическую схему по таблице истинности? По алгебраической формуле?

6. Что такое система счисления? Чем отличается позиционная система счисления от непозиционной?
7. Как получить качественный эквивалент числа в непозиционной системе счисления? В позиционной?
8. Как перевести числа из двоичной системы счисления в десятичную? Восьмеричную? Шестнадцатеричную? И наоборот?
9. Что такое двоично-десятичное число?
10. Как в ЭВМ представляются отрицательные числа и числа с плавающей запятой?
11. Что такое дополнительный код? Зачем он используется?
12. Как перевести десятичное число с плавающей запятой в двоичное?
13. Какие базовые типы данных используются для хранения переменных в языке СИ?
14. Что такое флаг? Зачем он используется? Каким образом можно манипулировать флагами? Что такое маска?