Лабораторная работа № 3

«Исследование шестнадцатеричного объектного формата файла»

Цель работы

Исследование текстового формата файла для хранения двоичных данных.

Краткая теоретическая справка

Шестнадцатеричный объектный формат файлов Intel-HEX (*англ. Hexadecimal Object File Format Specification*) (далее просто «прошивка») – это способ представления двоичные данные в виде кодов ASCII. Поскольку файл состоит из символов ASCII, а не двоичных кодов, появляется возможность хранить данные на бумаге, перфоленте или перфокартах, выводить их на терминал, принтер и т.д. Восьмибитовый HEX–формат файлов предусматривает размещение данных и кода в 16–разрядном линейном адресном пространстве для 8–разрядных процессоров Intel. 16-разрядный HEX-формат файлов дополнительно позволяет использовать 20-разрядное сегментное пространство адресов 16-разрядных процессоров Intel. И, наконец, 32-разрядный формат позволяет оперировать линейным 32-разрядным адресным пространством 32-разрядных процессоров.

Шестнадцатеричное представление двоичных данных в виде ASCII требует использование двух символов для записи одного байта, при этом первый символ всегда соответствует старшей тетраде битов одного байта. Такой подход увеличивает количество символов в двое по сравнению с количеством двоичных данных. Формат файла организован в виде набора записей, содержащих сведения о типе, количестве данных, адресе их загрузки в память и дополнительные сведения. В настоящее время определены 6 различных типов записей, однако не все их комбинации определены для разных форматов данных1. Записи могут быть следующих типов:

- '00' **Data Record** (запись, содержащая данные)
- '01' **End of File Record** (маркер конца файла)
- '02' **Extended Segment Address Record** (запись адреса расширенного сегмента)
- '03' **Start Segment Address Record** (запись адреса начала сегмента)
- '04' **Extended Linear Address Record** (запись расширенного линейного адреса)
- '05' **Start Linear Address Record** (Линейный адрес старта (определена только для 32-битного формата))

Для микроконтроллеров AVR в HEX-файле появляются только записи типа 00, 01 и 03. При этом запись типа 03 содержит абсолютный адрес старта программы в памяти, и не несет в себе никакого практического значения (эта запись обычно находится ближе к концу файла, перед записью 01 End of File Record).

Порядок данных в прошивке необычен для Intel: фактически хранение данных в памяти осуществляется с точностью «до наоборот», т.е. от **младшего байта к старшему**. Такой порядок даже называют «Интеловским».

Общий формат записи (General Record Format)

RECORD MARK ':'	**RECLEN**	**OFFSET**	**RECTYP**	**INFO или DATA**	**CHKSUM**
Маркер записи	Кол-во данных	Смещение	Тип записи	Данные или информация	Контрольная сумма
1 байт	1 байт	2 байта	1 байт	RECLEN байт	1 байт

Каждая запись представляет собой ASCII–строку файла. В одной строке – одна запись.

Каждая запись начинается с **МАРКЕРА ЗАПИСИ**, который обозначается ASCII-символом двоеточие («:»).

Каждая запись содержит поле **RECLEN**, определяющее количество байтов данных или информационных байтов, назначение которых определяется типом записи. Максимальное значение этого поля – 255 (0FF в шестнадцатеричном).

Каждая запись содержит поле **OFFSET**, определяющее 16-битное смещение в адресном пространстве байтов данных. Это поле используется только в записях данных, а в остальных случаях оно должно быть равно нулю.

Каждая запись содержит поле **TYPEREC**, определяющее тип текущей записи (из ранее упомянутых шести). Это поле используется для интерпретации всех остальных полей записи. Типы записей кодируются следующими значениями поля TYPEREC (в ASCII):

«00» – данные

«01» – маркер конца файла

«02» – адрес сегмента

«03» – сегментный адрес старта

«04» – линейный адрес

«05» – линейный адрес старта

Каждая запись содержит поле **INFO/DATA** переменной длины, которое содержит ноль или более байтов, закодированных символами ASCII. Назначение этих байтов определяется типом записи.

Наконец, каждая запись завершается полем **CHECKSUM**, гарантирующим целостность всех данных записи. Значение этого поля равно дополнению по модулю 256 до нуля суммы по модулю 256 всех байтов, начиная с поля RECLEN и заканчивая последним байтом поля **INFO/DATA**. При считывании записи следует суммировать по модулю 256 все байты записи, включая поле **CHECKSUM**. Если в конце концов сумма равна нулю, это означает, что данные считаны без искажений, в противном случае данные недостоверны.

Запись «Линейный адрес»

Формат записи, следующий:

RECORD MARK ':'	**RECLEN**	**OFFSET**	**RECTYP**	**INFO или DATA**	**CHKSUM**
Маркер записи	Кол-во данных	Смещение	Тип записи	Данные или информация	Контрольная сумма
«:»	«02»	«0000»	«04»	2 байта	1 байт

Эта запись служит для задания значения битов 16-31 в линейном базовом адресе (LBA, Linear Base Address), причем биты 0-15 LBA равны нулю. Биты 16-31 LBA определяются верхним линейным базовым адресом (ULBA, Upper Linear Base Address). Абсолютное значение адреса байта данных в памяти определяется как сумма значения LBA и значения поля OFFSET в последующих записях данных, плюс индекс байта данных внутри поля DATA. Эта сумма выполняется без учета переполнения результата (т.е. не может превышать 0FFFFFFFF, 4Гб). Фактический линейный адрес байта данных вычисляется в итоге по формуле:

ByteAddr = (LBA+DRLO+DRI) mod 4G, где

DRLO – значение поля OFFSET записи данных,

DRI – индекс байта в поле DATA записи данных,

mod 4G – операция «сложение по модулю 32».

Запись «**Адрес сегмента**»

Формат этой записи, следующий:

RECORD MARK ':'	**RECLEN**	**OFFSET**	**RECTYP**	**INFO или DATA**	**CHKSUM**
Маркер записи	Кол-во данных	Смещение	Тип записи	Данные или информация	Контрольная сумма
«:»	«02»	«0000»	«02»	2 байта	1 байт

Эта запись служит для задания значения битов 4-19 сегментного базового адреса (SBA, Segment Base Address), где биты 0-3 SBA равны нулю. Биты 4-19 SBA определяются верхним базовым адресом сегмента (USBA, Upper Segment Base Address). Абсолютный адрес байта в записи данных вычисляется путем прибавления к SBA значения поля OFFSET записи данных и индекса байта относительно начала поля DATA/INFO.

Прибавление смещения (OFFSET) осуществляется по модулю 65536 (64K), без учета переполнения. Таким образом, адрес конкретного байта вычисляется по формуле:

ByteAddr = SBA+ ([DRLO + DRI] mod 64K), где

DRLO – значение поля OFFSET записи данных,

DRI – индекс байта в поле DATA записи данных,

mod 64K – операция «сложение по модулю 65536».

Когда запись «Адрес сегмента» встречается в файле, вычисляется значение SBA, которое действует для всех последующих записей данных, пока не встретится снова запись «Адрес сегмента». По умолчанию SBA=0.

Запись данных

Формат этой записи, следующий:

RECORD MARK ':'	**RECLEN**	**OFFSET**	**RECTYP**	**INFO или DATA**	**CHKSUM**
Маркер записи	Кол-во данных	Смещение	Тип записи	Данные или информация	Контрольная сумма
«:»	1 байт	2 байта	«00»	**RECLEN** байта	1 байт

Данная запись содержит полезные данные. Метод вычисления фактического (абсолютного) адреса каждого байта данных в памяти определяется по вышеприведенным формулам и зависит от формата данных.

Назначение всех полей этой записи рассмотрено ранее.

Линейный адрес старта

Формат этой записи, следующий:

RECORD MARK ':'	RECLEN	OFFSET	RECTYP	INFO или DATA	CHKSUM
Маркер записи	Кол-во данных	Смещение	Тип записи	Данные или информация	Контрольная сумма
«:»	«04»	«0000»	«05»	4 байта	1 байт

Запись «Линейный адрес старта» используется для указания адреса, с которого начинается исполнение объектного файла. Это значение заносится в регистр EIP процессора. Обратите внимание, что эта запись определяет только точку входа сегмента кода для защищенного режима процессоров 803862. В обычном режиме точка старта определяется записью «Сегментный адрес старта», которая определяет значения пары регистров CS:IP.

Запись «Линейный адрес старта» может находиться в любом месте файла. Если ее нет, загрузчик использует адрес старта по умолчанию.

Значение регистра EIP процессора содержится в соответствующем поле записи, для него требуется всегда 4 байта. Назначение остальных полей записи рассмотрено ранее.

Сегментный адрес старта

Формат этой записи, следующий:

RECORD MARK ':'	RECLEN	OFFSET	RECTYP	INFO или DATA	CHKSUM
Маркер записи	Кол-во данных	Смещение	Тип записи	Данные или информация	Контрольная сумма
«:»	«04»	«0000»	«03»	4 байта	1 байт

Запись «Сегментный адрес старта» используется для указания адреса, с которого начинается исполнение объектного файла. Это значение определяет 20-битный адрес, заносимый в регистры CS:IP процессора. Обратите внимание, что эта запись определяет только точку входа в 20-битном адресном пространстве процессоров 8086/801863.

Запись «Сегментный адрес старта» может находиться в любом месте файла. Если ее нет, загрузчик использует значение по умолчанию.

Значение регистров CS:IP процессора содержится в соответствующем поле записи, для него требуется всегда 4 байта. Значение хранится в порядке «от старшего к младшему», т.е. младший байт значения регистра IP хранится в четвертом байте поля CS:IP, старший – в третьем, затем во втором хранится младший байт значения регистра CS, и в первом– старший байт регистра CS4.

Назначение остальных полей записи рассмотрено ранее.

Маркер конца файла (терминатор)

Формат этой записи, следующий:

RECORD MARK ':'	**RECLEN**	**OFFSET**	**RECTYP**	**INFO или DATA**	**CHKSUM**
Маркер записи	Кол-во данных	Смещение	Тип записи	Данные или информация	Контрольная сумма
«:»	«00»	«0000»	«01»	4 байта	1 байт

Пример задания

Логический элемент (входы C и E не задействованы) описанный в управляющей микропрограмме МК представлен на рисунке 1, программа, имитирующая его работу представлена в листинге 1. Сгенерированная прошивка представлена в листинге 2.

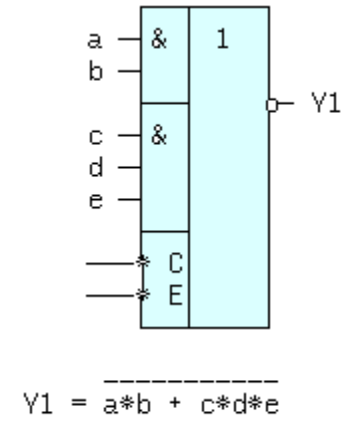

Рисунок 1 – Логические элемент И-ИЛИ

```
;
; HEX_FILE_LAB_3.asm
;
; Created: 28.11.2021 01:40:24
; Author : MYaro
;
// Вариант - DEMO
.include "m328pdef.inc"

// присваиваю мнемонические имена РОНам
```

```
.def   temp = r4                  // Вспомогательные временные РОНы
.def   temp2 = r8
.def   temp3 = r3
.def   temp4 = r2

.def   RN1 = r16                  //     Рабочие РОНы
.def   RN2 = r18                  //
.def   RN3    = r19               //
.def   RN4    = r29               //
.def   RN5    = r30               //

.def RESULT = r24

.equ ONE = 0xFF
.equ NULL = 0x00

.cseg
.org 0x06                         // Начальный адрес программы
start:
nop
// Начало программы
// Загрузка переменных в регистры

// Входные регистры
LDI RN1, ONE
LDI RN2, ONE
LDI RN3, ONE
LDI RN4, NULL
LDI RN5, ONE

//     LDI RN4, NULL
//     LDI RN3, ONE

// Начало логики работы программы
// Имитация элемента (2И + 3И) - ИЛИ - НЕ
Main:
nop
AND RN1, RN2               // Элемент 2И

AND RN3, RN4               // Элемент 3И
AND RN3, RN5

OR RN1, RN3                       // Элеент ИЛИ
COM RN1                                 // Элемент НЕ

MOV RESULT, RN1                   // Выходной регистр

infinity_loop:
nop                                     // Бесконечный цикл
       jmp infinity_loop
```

Листинг 1 – исходный код программы имитирующей поведение логического элемента И-ИЛИ-НЕ.

```
:020000020000FC
:10000C00000000FEF2FEF3FEFD0E0EFEF00000223E7
:10001C003D233E230302B00958028F00000C941300EE
:00000001FF
```

Листинг 2 – Сгенерированная прошивка

Пример парсинга строки прошивки

Дана строка с содержанием:

:10000C0000000FEF2FEF3FEFD0E0EFEF00000223E7

Эта строка требует записать в память FLASH микроконтроллера, начиная с адреса 0x000C (адрес указан в поле LOAD OFFSET) байты 00 00 0F EF 2F EF 3F EF D0 E0 EF EF 00 00 02 23 (байты указаны в поле DATA), всего 16 байт (так как в поле RECLEN указано значение 0x10). Тип записи – 0x00 RECTYP обозначает тип строки (записи) Data Record. При этом контрольная сумма CHKSUM записи (сумма байт полей RECLEN, LOAD OFFSET, RECTYP, DATA) равна E7 (дополнение до двух). Сумма байт всех полей RECLEN, LOAD OFFSET, RECTYP, DATA, CHKSUM должна быть равна 0.

:00000001FF

Строка – терминатор. Байт с число 01 сигнализирует о конце файла, полезных данных для загрузки в ПЗУ нет.

Задание

1. Исследовать шестнадцатеричный файл «прошивки» с расширением Intel HEX в соответствие со своим вариантов. Для подсветки полей файла рекомендуется использовать программу Notepad++.
2. Установить закодированные инструкции и их аргументы. Установить последовательность выполнения инструкции и по установленной последовательности восстановить закодированную логическую функцию. Определить начальный адрес программы в ПЗУ.
3. Определить регистры содержащие ВХОДНЫЕ данные (вход логической функции) и определить регистр содержащим ВЫХОДНЫЕ данные (выход логической функции).

Требования к содержанию отчета:

Выводы приводятся либо по каждому пункту программы работы, либо в конце отчета. Выводы должны быть конкретными и не содержать перечисление действий. Отчет в целом должен быть составлен таким образом, чтобы для понимания содержания и результатов проведенной работы не требовалось дополнительных устных пояснений.

Составление подобных отчетов – первый шаг к оформлению технических отчетов по экспериментальным исследованиям, которые предстоит проводить будущему специалисту.

Отчет должен содержать:

– титульный лист, содержание, цель работы, основную часть, выводы;

– информацию о варианте задания;

– вариант решения задания (восстановленная программа и логическая схема элемента, перечень используемых РОНов);

– выводы.