

Предисловие переводчика.

Перевод выполнен ARV для размещения на сайте <http://arvresearch.nm.ru>.

В переводе сохранен порядок описания, присущий оригинальному документу «*Intel Hexadecimal Object File. Format Specification. Revision A, 1/6/88*», хотя на взгляд переводчика, он не совсем логичный. Повторы, присущие оригинальному файлу, из перевода исключены.

В тексте перевода в кавычках («») принято показывать символы, без кавычек — числа и имена полей.

В таблицах, описывающих формат записей, в первой строке указано наименование поля, во второй — контекстное имя значения поля, в третьей приведено действительное значение (символы) или указано количество байтов, содержащихся в поле. Следует учитывать, что внутри Intel-HEX файла каждый *байт* записывается *двумя символами*.

Ввиду того, что перевод выполнен не профессиональным переводчиком, возможны неточности перевода отдельных терминов, специфичных для рассматриваемой темы, но смысловое содержание от этого пострадать не должно.

Настоящий документ не является официальной документацией Intel.

Авторское право на формат файла принадлежит Intel Corp.

Введение

Этот документ описывает формат объектного шестнадцатиричного файла Intel-HEX для 8-, 16- и 32-битных микроконтроллеров. Он может быть использован в качестве исходного для программаторов или аппаратных эмуляторов.

Шестнадцатиричный объектный формат файлов Intel-HEX (далее просто HEX-формат) – это способ представить 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.

Шестнадцатиричное 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:

- **Данные** (определена для всех форматов данных)
- **Маркер конца файла** (определена для всех форматов файла)
- **Сегментный адрес** (определена для 16- и 32-битных форматов)
- **Сегментный адрес старта** (определена для 16- и 32-битных форматов)
- **Линейный адрес** (определена только для 32-битного формата)
- **Линейный адрес старта** (определена только для 32-битного формата)

Общий формат записей

МАРКЕР ЗАПИСИ	КОЛ-ВО ДАННЫХ	СМЕЩЕНИЕ	ТИП ЗАПИСИ	ДАННЫЕ или ИНФОРМАЦИЯ	КОНТРОЛЬНАЯ СУММА
	RECLEN	OFFSET	TYPERECD	INFO/DATA	CHECKSUM
«:»	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 шестнадцатиричном).

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

- «00» – данные
- «01» – маркер конца файла
- «02» – адрес сегмента
- «03» – сегментный адрес старта
- «04» – линейный адрес
- «05» – линейный адрес старта

¹ Под форматом данных подразумевается разрядность шины данных процессора. Здесь и далее – примечания переводчика.

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:

МАРКЕР ЗАПИСИ	КОЛ-ВО ДАННЫХ	СМЕЩЕНИЕ	ТИП ЗАПИСИ	ДАнные или ИНФОРМАЦИЯ	КОНТРОЛЬНАЯ СУММА
	RECLen	OFFSET	TYPE/REC	ULBA	CHECKSUM
«:»	«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 0FFFFFFF, 4Гб). Фактический линейный адрес байта данных вычисляется в итоге по формуле:

$\text{ByteAddr} = (\text{LBA} + \text{DRLO} + \text{DRI}) \bmod 4\text{Г}$, где

DRLO – значение поля **OFFSET** записи данных,

DRI – индекс байта в поле **DATA** записи данных,

$\bmod 4\text{Г}$ – операция «сложение по модулю 32».

Когда запись «Линейный адрес» встречается в файле, вычисляется значение **LBA**, которое действует для всех последующих записей данных, пока не встретится снова запись «Линейный адрес». По умолчанию **LBA=0**.

Значение остальных полей рассмотрено ранее.

Запись «Адрес сегмента»

Формат этой записи следующий:

МАРКЕР ЗАПИСИ	КОЛ-ВО ДАННЫХ	СМЕЩЕНИЕ	ТИП ЗАПИСИ	ДАнные или ИНФОРМАЦИЯ	КОНТРОЛЬНАЯ СУММА
	RECLen	OFFSET	TYPE/REC	USBA	CHECKSUM
«:»	«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 (64К), без учета переполнения. Таким образом, адрес конкретного байта вычисляется по формуле:

$\text{ByteAddr} = \text{SBA} + ([\text{DRLO} + \text{DRI}] \bmod 64\text{К})$, где

DRLO – значение поля **OFFSET** записи данных,

DRI – индекс байта в поле **DATA** записи данных,

$\bmod 64\text{К}$ – 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**.

Значение остальных полей рассмотрено ранее.

Запись данных

Формат этой записи следующий:

МАРКЕР ЗАПИСИ	КОЛ-ВО ДАННЫХ	СМЕЩЕНИЕ	ТИП ЗАПИСИ	ДАННЫЕ или ИНФОРМАЦИЯ	КОНТРОЛЬНАЯ СУММА
	RECLEN	OFFSET	TYPEREC	ULBA	CHECKSUM
«:»	1 байт	2 байта	«00»	RECLEN байтов	1 байт

Эта запись собственно и содержит 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:

МАРКЕР ЗАПИСИ	КОЛ-ВО ДАННЫХ	СМЕЩЕНИЕ	ТИП ЗАПИСИ	ДАННЫЕ или ИНФОРМАЦИЯ	КОНТРОЛЬНАЯ СУММА
	RECLEN	OFFSET	TYPEREC	EIP	CHECKSUM
«:»	«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:

МАРКЕР ЗАПИСИ	КОЛ-ВО ДАННЫХ	СМЕЩЕНИЕ	ТИП ЗАПИСИ	ДАННЫЕ или ИНФОРМАЦИЯ	КОНТРОЛЬНАЯ СУММА
	RECLEN	OFFSET	TYPEREC	CS:IP	CHECKSUM
«:»	«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

² Очевидно, и более старших моделей.

³ На счет 80286 в официально документации ничего не сказано, но, очевидно, верно и для них.

значения регистра CS, и в первом— старший байт регистра CS⁴.

Назначение остальных полей записи рассмотрено ранее.

Маркер конца файла (терминатор)

Формат этой записи следующий:

МАРКЕР ЗАПИСИ	КОЛ-ВО ДАННЫХ	СМЕЩЕНИЕ	ТИП ЗАПИСИ	КОНТРОЛЬНАЯ СУММА
	RECLEN	OFFSET	TYPE REC	CHECKSUM
«:»	«00»	«0000»	«01»	«FF»

Эта запись не содержит полей с изменяющимися данными, поэтому выглядит всегда совершенно одинаково: «:00000001FF». Запись обозначает конец данных в файле и должна быть⁵.

Пример записи⁶

:06000300D293D2328161AC

Для рассмотрения в строку введены пробелы между значениями полей, реально в записи их нет.

:	06	0003	00	D293D2328161	AC
Маркер записи	Будет 6 байт данных	Смещение адреса пер- вого байта данных = 3	00 - запись данных	6 байтов данных: 0xD2, 0x93, 0xD2, 0x32, 0x81, 0x61	Контрольная сумма

⁴ Вообще-то, такой порядок 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».

⁵ Все последующие строки, если они есть в файле, игнорируются.

⁶ В оригинальном файле примера нет.