

# Intel HEX format Parser

## Group -H

Haran, Himanshu Kumar (448428)  
Nengwenani, Mpho Cavin (448385)  
Ravi Shankar Yadav (448416)

# Intel HEX

- Intel HEX is a file format that conveys binary information in ASCII text form.
- It is commonly used for programming microcontrollers, EPROMs, and other types of programmable logic devices.
- In a typical application, a compiler or assembler converts a program's source code (such as in C or assembly language) to machine code and outputs it into a HEX file.
- The HEX file is then imported by a programmer to "burn" the machine code into a EPROM, or is transferred to the target system for loading and execution.
- Originally the Intel Hex format was designed for a **16 bit address range (64kb)**.
- Later the file format was enhanced to accommodate larger files with a **20 bit address range (1Mb) and even 32 bit address range (4Gb)**.

# Format Specification

- Intel HEX consists of lines of ASCII text that are separated by line feed or carriage return characters or both.
- Each text line is called a record.
- Each text line contains hexadecimal characters that encode multiple binary numbers.
- The binary numbers may represent data, memory addresses, or other values, depending on their position in the line and the type and length of the line.

# Record Structure

All data lines are called records and contains the following fields:

:ccaaaarrdd...ss

[:] Every *line starts* with a colon (Hex value 3A).

[CC] *byte-count*. {2 digit value (1 byte)}, counting the actual data bytes in the record.

[aaaa] *address field* {4 digit (2 byte)} number representing the *first address* to be used by this record.

[rr] *Record type*. {2 digit value (1 byte)} indicating the record type.

[dd...] The *actual data of this record*. There can be 0 to 255 data bytes per record .

[ss] *Checksum*. {2 digit (1 byte)} checksum, to verify the record has no errors.

# Checksum Calculation & Verification

- Calculation
  - A record's checksum byte is the two's complement (negative) of the data checksum.

- Verification

:ccaaarrrdd...ss

- $cc + aaH + aaL + rr + \text{sum}(dd...) + ss = 0$

# Record Type

- There are ***5 record*** types defined:
- '**00**' = Data Record
- '**01**' = End Of File Record
- '**02**' = Extended Segment Address Record
- '**03**' = Start Segment Address Record
- '**04**' = Extended Linear Address Record
- '**05**' = Start Linear Address Record.

# Record Type 00

- Type '00' is the main record type.
- The real data are sent using this record type.
- The 1st data byte of the record is stored in the address specified by the address field of the record (plus the *pre-set Segment* or *Linear Base Address*).

:10010000214601360121470136007EFE09D2190140

# Record Type 01

- Type '01' is the End Of File record.
- The receiver of the file will stop waiting for new records after receiving this record.

## Example

:00000001FF



## Record Type 02

- These records are used to pre-set the Extended Segment Address.
- With this segment address it is possible to send files of up to 1Mb in length.
- The Segment address is multiplied by 16 and then added to all subsequent address fields of type '00' records to obtain the effective address.

### Example

:020000020200FA

:10010000214601360121470136007EFE09D2190140

Address = 02000 + 0100 = 02100

## Record Type 03

- This record type is used to specify the start address for Intel processors, like the 8086.
- This starting address will be loaded into the CS and IP registers of the processor.

### Example

:040000030200013ABC

CS Register = 0200

IP Register = 013A

# Record Type 04

- Type '04' records are used to pre-set the Linear Base Address to obtain a full 32 bit address range.
- The Linear Base Address is used as the upper 16 bits in the 32 bit linear address space.
- The lower 16 bits will come from the address field of type '00' records.

## Example

:02000004007F7B

:10010000214601360121470136007EFE09D2190140

Address = 007F000 + 0100 = 007F0100

## Record Type 05

- This record type is used for Intel processors, like the 80386.
- Specify value of EIP register

### Example

:040000050200013ABA

EIP = 0200013A

# Input and Output

- ***Input***

- ***I8HEX*** files use only record types 00 and 01 (16 bit addresses)
- ***I16HEX*** files use only record types 00 through 03 (20 bit addresses)
- ***I32HEX*** files use only record types 00, 01, 04, and 05 (32 bit addresses)

- **Output**

- Validation (format, address & check sum)
- Blocks of Address
- Information for each type of records

# Challenges

- Verification of valid address ranges.
  - Different address calculation for IHEX8, IHEX16, IHEX32
  - Address are not always incrementing linearly

# Thanks

Thanks for listening!

Questions?