4 The BAM Format Specification

4.1 The BGZF compression format

BGZF is block compression implemented on top of the standard gzip file format.[26] The goal of BGZF is to provide good compression while allowing efficient random access to the BAM file for indexed queries. The BGZF format is 'gunzip compatible', in the sense that a compliant gunzip utility can decompress a BGZF compressed file.[27]

A BGZF file is a series of concatenated BGZF blocks, each no larger than ~~64Kb before or~~ 64KB before and after compression. Each BGZF block is itself a spec-compliant gzip archive which contains an "extra field" in the format described in RFC1952. The gzip file format allows the inclusion of application-specific extra fields and these are ignored by compliant decompression implementation. The gzip specification also allows gzip files to be concatenated. The result of decompressing concatenated gzip files is the concatenation of the uncompressed data.

Each BGZF block contains a standard gzip file header with the following standard-compliant extensions:

1. The F.EXTRA bit in the header is set to indicate that extra fields are present.
2. The extra field used by BGZF uses the two subfield ID values 66 and 67 (ASCII 'BC').
3. The length of the BGZF extra field payload (field LEN in the gzip specification) is 2 (two bytes of payload).
4. The payload of the BGZF extra field is a 16-bit unsigned integer in little endian format. This integer gives the size of the containing BGZF block minus one.

On disk, a complete BGZF file is a series of blocks as shown in the following table. (All integers are little endian as is required by RFC1952.)

Field	Description	Type	Value
List of compression blocks (until the end of the file)			
ID1	gzip IDentifier1	`uint8_t`	31
ID2	gzip IDentifier2	`uint8_t`	139
CM	gzip Compression Method	`uint8_t`	8
FLG	gzip FLaGs	`uint8_t`	4
MTIME	gzip Modification TIME	`uint32_t`	
XFL	gzip eXtra FLags	`uint8_t`	
OS	gzip Operating System	`uint8_t`	
XLEN	gzip eXtra LENgth	`uint16_t`	
Extra subfield(s) (total size=XLEN)			
Additional RFC1952 extra subfields if present			
SI1	Subfield Identifier1	`uint8_t`	66
SI2	Subfield Identifier2	`uint8_t`	67
SLEN	Subfield LENgth	`uint16_t`	2
BSIZE	total Block SIZE minus 1	`uint16_t`	
Additional RFC1952 extra subfields if present			
CDATA	Compressed DATA by zlib::deflate()	`uint8_t[BSIZE-XLEN-19]`	
CRC32	CRC-32	`uint32_t`	
ISIZE	Input SIZE (length of uncompressed data)	`uint32_t`	

The random access method to be described next limits the uncompressed contents of each BGZF block to a maximum of 2^{16} bytes of data. Thus while ISIZE is stored as a `uint32_t` as per the gzip format, in BGZF it is limited to the range $[0, 65536]$. BSIZE can represent BGZF block sizes in the range $[1, 65536]$, though typically BSIZE will be rather less than ISIZE due to compression.

[26] L. Peter Deutsch, **GZIP file format specification version 4.3**, *RFC 1952*.

[27] It is worth noting that there is a known bug in the Java GZIPInputStream class that concatenated gzip archives cannot be successfully decompressed by this class. BGZF files can be created and manipulated using the built-in Java util.zip package, but naive use of GZIPInputStream on a BGZF file will not work due to this bug.