IBJ Format Working-Notes 2v0

John D. McMullin

31 September 2023

Note: This document is not intended to be a complete formal description of IBJ format

Contents

[1. Philosophy 3](#_Toc144756823)

[2. File Layout 4](#_Toc144756824)

[3. Data Types 5](#_Toc144756825)

[4. IBJ RECORDS 6](#_Toc144756826)

[3.1. OBJ 6](#_Toc144756827)

[3.2. DATA 6](#_Toc144756828)

[3.3. CONST 6](#_Toc144756829)

[3.4. DISPLAY 6](#_Toc144756830)

[3.5. JUMP 6](#_Toc144756831)

[3.6. JCOND 6](#_Toc144756832)

[3.7. CALL 7](#_Toc144756833)

[3.8. LABEL 7](#_Toc144756834)

[3.9. FIXUP 7](#_Toc144756835)

[3.10. SETFIX 7](#_Toc144756836)

[3.11. REQEXT 7](#_Toc144756837)

[3.12. REFLABEL 8](#_Toc144756838)

[3.13. REFEXT 8](#_Toc144756839)

[3.14. BSS 8](#_Toc144756840)

[3.15. COTWORD 8](#_Toc144756841)

[3.16. DATWORD 8](#_Toc144756842)

[3.17. SWTWORD 8](#_Toc144756843)

[3.18. SOURCE 8](#_Toc144756844)

[3.19. DEFEXTCODE 9](#_Toc144756845)

[3.20. DEFEXTDATA 9](#_Toc144756846)

[3.21. SWT 9](#_Toc144756847)

[3.22. LINE 9](#_Toc144756848)

[3.23. ABSEXT 9](#_Toc144756849)

[Appendix 1. Example IBJ file 10](#_Toc144756850)

# Philosophy

The IBJ format is a textual representation used to provide an interface between the code generator stage and the Object file generator stage of a compiler.

It is an Instruction Set Architecture neutral format, aimed at describing how to generate the Object file contents. The same IBJ file can be used to generate ELF or COFF object files.

# File Layout

Each line of an IBJ file represents an IBJ record encoded using ASCII characters.

An IBJ record contains a array sequence of ASCII characters [1..N] starting with an ASCII character in the range: (‘A’ .. ‘W’) used to indicate the IBJ record type.

So IBJ record type is indicated by character[1].

Characters[2..N] are encoded hexadecimal numbers stored as ASCII (‘0’..’9’,’A’..’F’ or ’a’..’f’) in pairs. Each pair of characters represents a byte value.

Characters [2..3] form a byte count of the number of following Hex characters, where:

Character[2] is the high nibble value (as a Hex value).

Character[3] is the low nibble value (as a Hex value).

This limits the number of subsequent bytes to a maximum of 255 after the count byte.

Thus, the first Hex character containing specific data about the IBJ record info is character[4] where the format of the following bytes depends on the IBJ record type.

|  |  |  |
| --- | --- | --- |
| **Record Character** | **IBJ Type** | **IBJ Purpose** |
| A | IF OBJ | plain object code |
| B | IF DATA | data seg offset code word |
| C | IF CONST | const seg offset code word |
| D | IF DISPLAY | display seg offset code word |
| E | IF JUMP | unconditional jump to label |
| F | IF JCOND | cond jump to label JE, JNE, JLE, JL, JGE, JG |
| G | IF CALL | call a label |
| H | IF LABEL | define a label |
| I | IF FIXUP | define location for stack fixup instruction |
| J | IF SETFIX | stack fixup <location> <amount> |
| K | IF REQEXT | external name spec |
| L | IF REFLABEL | relative address of label |
| M | IF REFEXT | external name relative offset code word (call external) |
| N | IF BSS | BSS segment offset code word |
| O | IF COTWORD | Constant table word |
| P | IF DATWORD | Data segment word |
| Q | IF SWTWORD | switch table entry - actually a label ID |
| R | IF SOURCE | name of the source file |
| S | IF DEFEXTCODE | define a code label that is external |
| T | IF DEFEXTDATA | define a data label that is external |
| U | IF SWT | switch table offset code word |
| V | IF LINE | line number info for debugger |
| W | IF ABSEXT | external name absolute offset code word (data external) |

# Data Types

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data-type** | **Size** | | **Format** | | |
| **Hex** | **Bytes** | **H** | **Hex Char** | **‘0’..’9’,’A’..’F’ or ‘a’..’f’ as ASCII char** |
| **B** | **Byte Value** | **0..255** |
| **W** | **16 bit Word value** | **0..65536** |
| **A** | **8-bit ASCII Character** | |
| Byte | 2 | 1 | HH = B  H[1] = High value nibble  H[2] = Low value nibble | | |
| Condition | 2 | 1 | HH = B  B = ? | | |
| LabelNo | 4 | 2 | HHHH = BB = W  H[1]H[2] = High value byte  H[3]H[4] = Low value byte | | |
| ShortInt | 4 | 2 | HHHH = BB = W  H[1]H[2] = High value byte  H[3]H[4] = Low value byte | | |
| HexString | N | N/2 | HH..HH = B..B  This forms a sequence of byte values where each byte is in the range 0..255. | | |
| NameString | N | N/2 | HH..HH = A..A  This forms a sequence of ASCII characters or a string  A[1] the first ASCII character  A[N/2] the last ASCII character | | |

# IBJ RECORDS

## OBJ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘A’ | Data | | HexString |
| Effect: |  | | | |
| Notes: | plain object code (in binary format) | | | |
| Error: |  | | | |
| Example: | A020FBFC0 | | As assembler text  MOVSX %EAX,%AX | |

## DATA

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘B’ | Data | | ShortInt |
| Effect: |  | | | |
| Notes: | data seg offset code word | | | |
| Error: |  | | | |
| Example: |  | |  | |

## CONST

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘C’ | Data | | ShortInt |
| Effect: |  | | | |
| Notes: | const seg offset code word | | | |
| Error: |  | | | |
| Example: |  | |  | |

## DISPLAY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘D’ | Data | | ShortInt |
| Effect: |  | | | |
| Notes: | display seg offset code word | | | |
| Error: |  | | | |
| Example: |  | |  | |

## JUMP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘E’ | TargetLabel | | LabelNo |
| Effect: |  | | | |
| Notes: | unconditional jump to label | | | |
| Error: |  | | | |
| Example: |  | |  | |

## JCOND

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘F’ | JumpCondition | | Condition |
| TargetLabel | | LabelNo |
| Effect: |  | | | |
| Notes: | cond jump to label JE, JNE, JLE, JL, JGE, JG | | | |
| Error: |  | | | |
| Example: |  | |  | |

## CALL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘G’ | TargetLabel | | LabelNo |
| Effect: |  | | | |
| Notes: | call a label | | | |
| Error: |  | | | |
| Example: |  | |  | |

## LABEL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘H’ | LabelDef | | LabelNo |
| Effect: |  | | | |
| Notes: | define a label | | | |
| Error: |  | | | |
| Example: |  | |  | |

## FIXUP

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction: | ‘I’ | FixupId | ShortInt |
| StringSize | Byte |
| NameString | NameString |
| Effect: |  | | |
| Notes: | define location for stack fixup instruction | | |
| Error: |  | | |
| Example: |  |  | |

## SETFIX

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction: | ‘J’ | FixupId | ShortInt |
| Offset | ShortInt |
| Events | ShortInt |
| Trap | ShortInt |
| From | ShortInt |
| Effect: |  | | |
| Notes: | stack fixup <location> <amount> | | |
| Error: |  | | |
| Example: |  |  | |

## REQEXT

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction: | ‘K’ | ExternalName | NameString |
| Effect: |  | | |
| Notes: | external name spec | | |
| Error: |  | | |
| Example: |  |  | |

## REFLABEL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘L’ | LabelRefNo | | ShortInt |
| Offset | | ShortInt |
| Effect: |  | | | |
| Notes: | relative address of label | | | |
| Error: |  | | | |
| Example: |  | |  | |

## REFEXT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘M’ | ExtNameRef | | ShortInt |
| Offset | | ShortInt |
| Effect: |  | | | |
| Notes: | external name relative offset code word (call external) | | | |
| Error: |  | | | |
| Example: |  | |  | |

## BSS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘N’ | Offset | | ShortInt |
| Effect: |  | | | |
| Notes: | BSS segment offset code word | | | |
| Error: |  | | | |
| Example: |  | |  | |

## COTWORD

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘O’ | Data | | ShortInt |
| Effect: |  | | | |
| Notes: | Constant table word | | | |
| Error: |  | | | |
| Example: |  | |  | |

## DATWORD

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘P’ | Data | | ShortInt |
| Effect: |  | | | |
| Notes: | Data segment word | | | |
| Error: |  | | | |
| Example: |  | |  | |

## SWTWORD

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘Q’ | SwitchLabelNo | | ShortInt |
| Effect: |  | | | |
| Notes: | switch table entry - actually a label ID | | | |
| Error: |  | | | |
| Example: |  | |  | |

## SOURCE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘R’ | FileName | | NameString |
| Effect: |  | | | |
| Notes: | name of the source file | | | |
| Error: |  | | | |
| Example: |  | |  | |

## DEFEXTCODE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘S’ | ExtCodeName | | NameString |
| Effect: |  | | | |
| Notes: | define a code label that is external | | | |
| Error: |  | | | |
| Example: |  | |  | |

## DEFEXTDATA

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘T’ | ExtDataName | | NameString |
| Effect: |  | | | |
| Notes: | define a data label that is external | | | |
| Error: |  | | | |
| Example: |  | |  | |

## SWT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘U’ | TableOffset | | ShortInt |
| Effect: |  | | | |
| Notes: | switch table offset code word | | | |
| Error: |  | | | |
| Example: |  | |  | |

## LINE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘V’ | LineNo | | ShortInt |
| Effect: |  | | | |
| Notes: | line number info for debugger | | | |
| Error: |  | | | |
| Example: |  | |  | |

## ABSEXT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction: | ‘W’ | ExtNameRefNo | | ShortInt |
| Offset | | ShortInt |
| Effect: |  | | | |
| Notes: | external name absolute offset code word (data external) | | | |
| Error: |  | | | |
| Example: |  | |  | |

1. Example IBJ file

This file has had excess IBJ records removed but is still a legal IBJ file.

R0962696C626F2E696D70

V020500

H02E803

S05787A65726F

I0C0000015A45524F434F554E54

V020600

A02C705

B0400000000

A0400000000

V020700

A02C9C3

J0A00000000000000000000

V020900

H02E903

S0478696E63

I0C100001494E4352454D454E54

V020A00

A02FF05

B0400000000

V020B00

A02C9C3

J0A10000000000000000000

V020D00

H02EA03

S0478646563

I0C1C000144454352454D454E54

V020E00

A02FF05

B0400000000

V020F00

A02C9C3

J0A1C000000000000000000

V021100

H02EB03

S067876616C7565

I0828000156414C5545

V021200

A01A1

B0400000000

A02C9C3

V021300

J0A28000000000000000000

V021500

P020000

P020000