
# WebAssembly Code Metadata Specification

**WebAssembly Community Group**

**Andreas Rossberg (editor)**

**Mar 26, 2024**

## Contents



## 1 Introduction

This document defines a generic mechanism for attaching arbitrary metadata to WebAssembly instructions. Additionally, it defines specific metadata formats using this mechanism.

Such metadata do not contribute to, or otherwise affect, the WebAssembly semantics, and may be ignored by an implementation.

However, it can provides useful information that implementations can make use of to improve user experience or take compilation hints.

## 1.1 Dependencies

This document is based on the WebAssembly core specification (https://webassembly.github.io/spec/core/), and makes use of terms and definitions described there. These uses always link to the corresponding definition in the core specification.

# 2 Structure

## 2.1 Code Metadata

A Code Metadata item is a piece of information logically attached to an instruction.

An item is associated with a format, which defines the item's payload.

TODO: can this be expressed with the math notation?

### Branch Hints

A Branch Hint is a Code Metadata format that can be attached to *br_if* and *if* instructions. Its payload indicates whether the branch is likely or unlikely to be taken.

TODO: math definition

# 3 Binary Format

## 3.1 Code Metadata

All code metadata items of a given format $T$ are grouped under a custom section named *'metadata.code.T'*. The following parametrized grammar rules define the generic structure of a code metadata section of format $T$.

$$
\begin{array}{lcll}
\texttt{codemetadatasec}(\texttt{T}) & ::= & \texttt{section}_0(\texttt{codemetada}(\texttt{T})) & \\
\texttt{codemetada}(\texttt{T}) & ::= & n{:}\texttt{name} & (\text{if } n = \text{`}\texttt{metadata.code.T}\text{'}) \\
 & & \texttt{vec}(\texttt{codemetadafunc}(\texttt{T})) & \\
\texttt{codemetadafunc}(\texttt{T}) & ::= & fidx{:}\texttt{funcidx}\ \texttt{vec}(\texttt{codemetadataitem}(\texttt{T})) & \\
\texttt{codemetadataitem}(\texttt{T}) & ::= & instoff{:}\texttt{u32}\ \ size{:}\texttt{u32} & (\text{if } size = ||\texttt{T}||) \\
 & & data{:}\texttt{T} &
\end{array}
$$

Where *funcpos* is the byte offset of the annotation starting from the beginning of the function body, and *data* is a further payload, whose content depends on the format $T$.

*code metadata function* entries must appear in order of increasing *function id*, and duplicate id values are not allowed. *code metadata item* entries must appear in order of increasing *instruction offset*, and duplicate offset values are not allowed.

### Branch Hints

A Branch Hint is code metadata item with format *branch_hint*. All branch hints for a module are contained in a single code metadata section with name *'metadata.code.branch_hint'*.

$$
\begin{array}{lcl}
\texttt{branchhintsec} & ::= & \texttt{codemetadatasec}(\texttt{branchhint}) \\
\texttt{branchhint} & ::= & \texttt{0x00} \\
 & | & \texttt{0x01}
\end{array}
$$

# 4 Text Format

## 4.1 Code Metadata

Code Metadata items appear in the text format as custom annotations, and are considered attached to the first instruction that follows them.

$$\texttt{codemetadataannot}(\texttt{T}) \quad ::= \quad \text{`}(\texttt{@metadata.code.T}\text{'}\ \mathit{data}{:}\texttt{T}\ \text{`})\text{'}$$

Where $T$ is the format of the item, and *data* is a byte string containing the same payload as in the binary format.

# Index