# Updated GCC IFUNC and FMV Documentation

#### Introduction

This document provides comprehensive guidelines on using GCC IFUNC (Indirect Functions) and FMV (Function Multi-Versioning) across different architectures. It includes an overview, usage instructions, architecture-specific details, best practices, and examples.

#### IFUNC (Indirect Functions)

**Overview:** IFUNC allows the selection of different function implementations at runtime based on various criteria like CPU capabilities, optimizing performance by choosing the most suitable function for the current environment.

**Usage:** To use IFUNC, declare a function with the ifunc attribute and provide a resolver function to select the appropriate implementation.

**Syntax:**

*\_\_attribute\_\_((ifunc("resolver\_function")))*

**Example:**

*#include <stdio.h>*

*extern int foo(void) \_\_attribute\_\_((ifunc("foo\_resolver")));*

*int foo\_impl1(void) { return 1; }*

*int foo\_impl2(void) { return 2; }*

*void \*foo\_resolver(void) {*

*if (/\* condition \*/) {*

*return foo\_impl1;*

*} else {*

*return foo\_impl2;*

*}*

*}*

*int main(void) {*

*printf("%d\n", foo());*

*return 0;*

*}*

**Architecture-Specific Details:**

* **x86 Architecture:**
  + Syntax for manual specification: \_\_attribute\_\_((target("feature")))
  + Example:

*extern int foo(void) \_\_attribute\_\_((ifunc("foo\_resolver")));*

*int foo\_sse(void) { return 1; }*

*int foo\_avx(void) { return 2; }*

*void \*foo\_resolver(void) {*

*if (/\* check for AVX support \*/) {*

*return foo\_avx;*

*} else {*

*return foo\_sse;*

*}*

*}*

*int main(void) {*

*printf("%d\n", foo());*

*return 0;*

*}*

* + Ensure the resolver function is efficient as it impacts performance.
* **ARM Architecture:**
  + Syntax for manual specification: \_\_attribute\_\_((target\_version("feature")))
  + Example:

*extern int foo(void) \_\_attribute\_\_((ifunc("foo\_resolver")));*

*int foo\_neon(void) { return 1; }*

*int foo\_vfp(void) { return 2; }*

*void \*foo\_resolver(void) {*

*if (/\* check for NEON support \*/) {*

*return foo\_neon;*

*} else {*

*return foo\_vfp;*

*}*

*}*

*int main(void) {*

*printf("%d\n", foo());*

*return 0;*

*}*

* + Optimize the resolver for minimal overhead.

**Best Practices:**

* Ensure resolver efficiency to minimize performance overhead.
* Use IFUNC for performance-critical paths where different implementations can significantly improve performance.

#### FMV (Function Multi-Versioning)

**Overview:** FMV automates the process of creating multiple versions of functions optimized for different hardware features.

**Usage:** FMV can be specified manually or through automatic cloning, providing different versions of functions for various architectures.

**Syntax:**

* For manual specification: \_\_attribute\_\_((target("nnn")))
* For cloning: \_\_attribute\_\_((target\_clone("nnn1", "nnn2" [...])))

**Architecture-Specific Details:**

* **x86 Architecture:**
  + Example: \_\_attribute\_\_((target("sse4.2"))) int foo(void);
* **ARM Architecture:**
  + Example: \_\_attribute\_\_((target\_version("sve"))) int foo(void);

**Best Practices:**

* Align data structures to natural boundaries to maximize performance.
* Minimize unnecessary floating-point moves to reduce overhead.

#### Documentation Updates

* **GCC IFUNC Documentation:**
  + Updated syntax examples for x86 and ARM architectures.
  + Added best practices for resolver function efficiency and performance-critical paths.
* **FMV Documentation:**
  + Expanded usage instructions for both manual and cloned versions.
  + Included architecture-specific details for x86 and ARM.
  + Provided best practices for data alignment and reducing overhead.

**References:**

* [GCC Documentation](https://gcc.gnu.org/onlinedocs/)
* [GCC Git Repository](https://gcc.gnu.org/git.html)
* Community Forums for additional insights and support.

**Version History:**

* **Version 1.0:** Initial release.
* **Version 1.1:** Updated architecture-specific details and examples.