GCC中的intrinsic —— Built-In Functions

廖仕华 PLCT LAB
shihua@iscas.ac.cn

2023/12/15

主要内容

- 什么是Built-In Function
- 从零开始添加一个Builtin Function

参考资料：GCC Manual、GCC Internal Manual

是什么

- 什么是Built-In Function

在GCC中，有如下类型的 Built-In Function：

BUILT_IN_FRONTEND：ada,c,cp,rust等

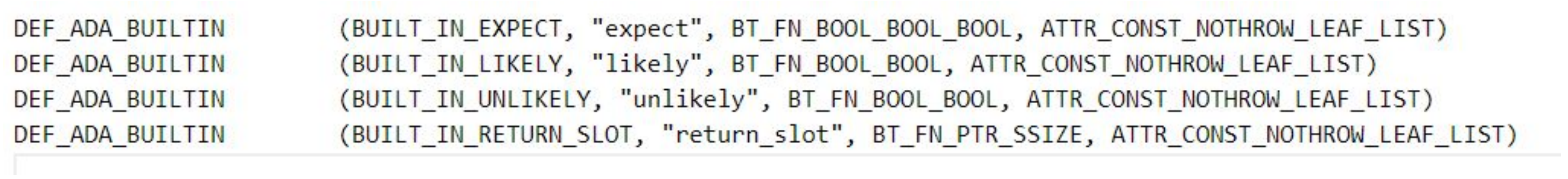
BUILT_IN_MD: i386,aarch64等

BUILT_IN_NORMAL：c99,libgcc等

是什么

- 什么是Built-In Function

/gcc/ada/gcc-interface/ada-builtin.def

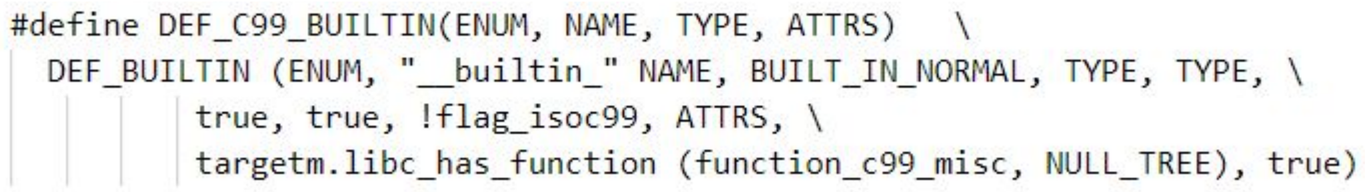


```
DEF_ADA_BUILTIN         (BUILT_IN_EXPECT, "expect", BT_FN_BOOL_BOOL_BOOL, ATTR_CONST_NOTHROW_LEAF_LIST)
DEF_ADA_BUILTIN         (BUILT_IN_LIKELY, "likely", BT_FN_BOOL_BOOL, ATTR_CONST_NOTHROW_LEAF_LIST)
DEF_ADA_BUILTIN         (BUILT_IN_UNLIKELY, "unlikely", BT_FN_BOOL_BOOL, ATTR_CONST_NOTHROW_LEAF_LIST)
DEF_ADA_BUILTIN         (BUILT_IN_RETURN_SLOT, "return_slot", BT_FN_PTR_SSIZE, ATTR_CONST_NOTHROW_LEAF_LIST)
```

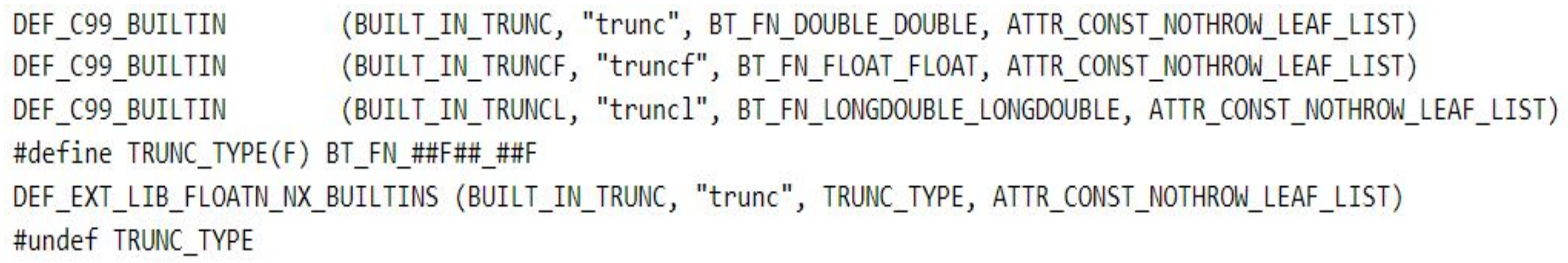
是什么

- 什么是Built-In Function

/gcc/gcc/builtin.def



```
#define DEF_C99_BUILTIN(ENUM, NAME, TYPE, ATTRS)   \
  DEF_BUILTIN (ENUM, "__builtin_" NAME, BUILT_IN_NORMAL, TYPE, TYPE, \
              true, true, !flag_isoc99, ATTRS, \
              targetm.libc_has_function (function_c99_misc, NULL_TREE), true)
```

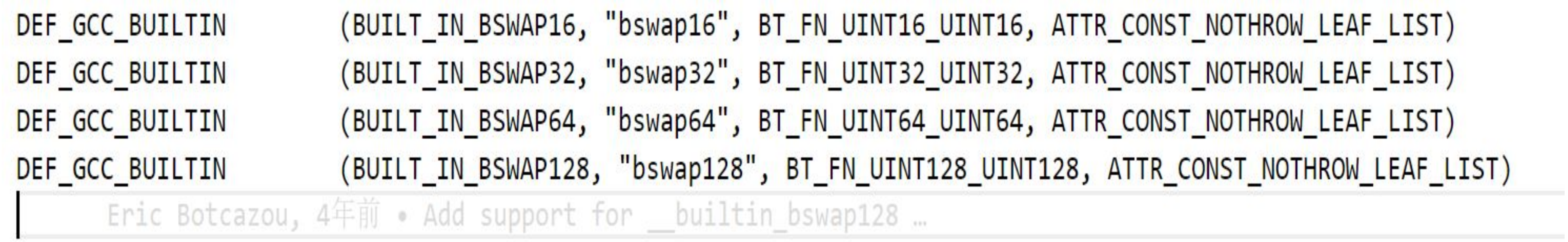


```
DEF_C99_BUILTIN        (BUILT_IN_TRUNC, "trunc", BT_FN_DOUBLE_DOUBLE, ATTR_CONST_NOTHROW_LEAF_LIST)
DEF_C99_BUILTIN        (BUILT_IN_TRUNCF, "truncf", BT_FN_FLOAT_FLOAT, ATTR_CONST_NOTHROW_LEAF_LIST)
DEF_C99_BUILTIN        (BUILT_IN_TRUNCL, "truncl", BT_FN_LONGDOUBLE_LONGDOUBLE, ATTR_CONST_NOTHROW_LEAF_LIST)
#define TRUNC_TYPE(F) BT_FN_##F##_##F
DEF_EXT_LIB_FLOATN_NX_BUILTINS (BUILT_IN_TRUNC, "trunc", TRUNC_TYPE, ATTR_CONST_NOTHROW_LEAF_LIST)
#undef TRUNC_TYPE
```

是什么

- 什么是Built-In Function

/gcc/gcc/builtin.def



```
DEF_GCC_BUILTIN        (BUILT_IN_BSWAP16, "bswap16", BT_FN_UINT16_UINT16, ATTR_CONST_NOTHROW_LEAF_LIST)
DEF_GCC_BUILTIN        (BUILT_IN_BSWAP32, "bswap32", BT_FN_UINT32_UINT32, ATTR_CONST_NOTHROW_LEAF_LIST)
DEF_GCC_BUILTIN        (BUILT_IN_BSWAP64, "bswap64", BT_FN_UINT64_UINT64, ATTR_CONST_NOTHROW_LEAF_LIST)
DEF_GCC_BUILTIN        (BUILT_IN_BSWAP128, "bswap128", BT_FN_UINT128_UINT128, ATTR_CONST_NOTHROW_LEAF_LIST)
        Eric Botcazou, 4年前 • Add support for __builtin_bswap128 ...
```

是什么

- 什么是Built-In Function

/gcc/gcc/builtin.cc

```
static rtx expand_builtin_bswap (machine_mode, tree, rtx, rtx)
```

/gcc/gcc/optabs.cc

```
static rtx widen_bswap (scalar_int_mode, rtx, rtx)
static rtx expand_doubleword_bswap (machine_mode, rtx, rtx)
```

是什么

- **什么是**Built-In Function

  前文提及， BUILT_IN_MD **是与架构相关的** Built-in function,**最终生成的是具体的指令。**

  以i386**为例，** /gcc/gcc/config/i386/ia32intrin.h

```
extern __inline int
__attribute__((__gnu_inline__, __always_inline__,
__artificial__))
__bsrsi (int __X)
{
  return __builtin_ia32_bsrsi (__X);
}
```

bsr eax

是什么

- 什么是Built-In Function
  /gcc/gcc/config/i386.md

```
(define_insn "bsr"
  [(set (reg:CCZ FLAGS_REG)
  (compare:CCZ (match_operand:SI 1 "nonimmediate_operand"
"rm")
             (const_int 0)))
   (set (match_operand:SI 0 "register_operand" "=r")
  (minus:SI (const_int 31)
        (clz:SI (match_dup 1))))]
  ""
  "bsr{l}\t{%1, %0|%0, %1}"
```

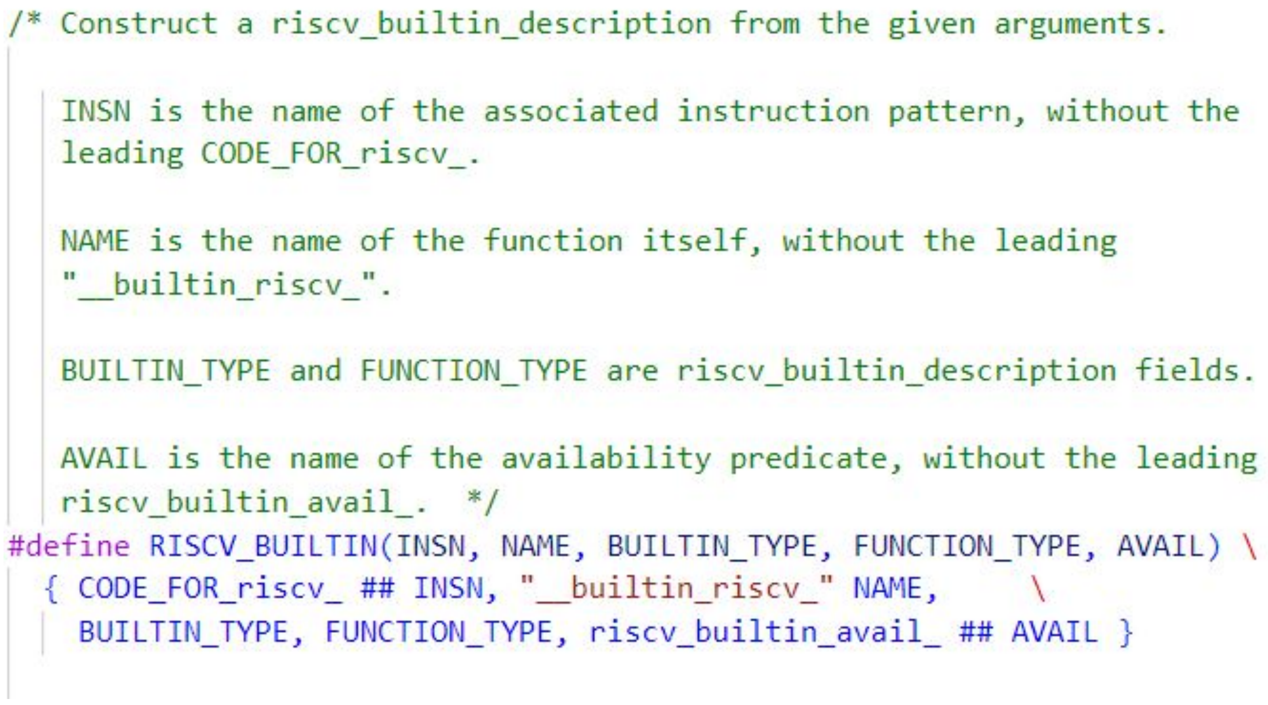
是什么

- 什么是Built-In Function
  /gcc/gcc/config/i386.def

```
BDESC_FIRST (... CODE_FOR_bsr, "__builtin_ia32_bsrsi",
IX86_BUILTIN_BSRSI,...(int) INT_FTYPE_INT)
```

怎么添加

以riscv为例



```
/* Construct a riscv_builtin_description from the given arguments.

   INSN is the name of the associated instruction pattern, without the
   leading CODE_FOR_riscv_.

   NAME is the name of the function itself, without the leading
   "__builtin_riscv_".

   BUILTIN_TYPE and FUNCTION_TYPE are riscv_builtin_description fields.

   AVAIL is the name of the availability predicate, without the leading
   riscv_builtin_avail_.  */
#define RISCV_BUILTIN(INSN, NAME, BUILTIN_TYPE, FUNCTION_TYPE, AVAIL) \
  { CODE_FOR_riscv_ ## INSN, "__builtin_riscv_" NAME,      \
    BUILTIN_TYPE, FUNCTION_TYPE, riscv_builtin_avail_ ## AVAIL }
```

怎么添加

以riscv为例

INSN

```
(define_insn "riscv_xperm4_<mode>"
  [(set (match_operand:X 0 "register_operand" "=r")
        (unspec:X [(match_operand:X 1 "register_operand" "r")
                   (match_operand:X 2 "register_operand" "r")]
                   UNSPEC_XPERM4))]
  "TARGET_ZBKX"
  "xperm4\t%0,%1,%2"
  [(set_attr "type" "crypto")])
```

怎么添加

以riscv为例

NAME

[riscv-c-api-doc/riscv-c-api.md at master · riscv-non-isa/riscv-c-api-doc (github.com)]()

Scalar Bit Manipulation Extension Intrinsics

```
uint64_t __riscv_xperm8_64(uint64_t rs1, uint64_t rs2); xperm8  Zbkx
```

怎么添加

以riscv为例

BUILTIN_TYPE

```
enum riscv_builtin_type {
  /* The function corresponds directly to an .md pattern.  */
  RISCV_BUILTIN_DIRECT,

  /* Likewise, but with return type VOID.  */
  RISCV_BUILTIN_DIRECT_NO_TARGET
};
```

怎么添加

以riscv为例

FUNCTION_TYPE

/gcc/gcc/config/riscv/riscv-ftype.def

```
DEF_RISCV_FTYPE (2, (UDI, UDI, UDI))
```

怎么添加

以riscv为例

AVAIL

AVAIL (crypto_zbkx64, TARGET_ZBKX && TARGET_64BIT)

怎么添加

以riscv为例

```
RISCV_BUILTIN (xperm8_di, "xperm8", RISCV_BUILTIN_DIRECT,
RISCV_UDI_FTYPE_UDI_UDI, crypto_zbkx64),
```

怎么添加

以riscv为例

```
RISCV_BUILTIN (xperm8_di, "xperm8", RISCV_BUILTIN_DIRECT,
RISCV_UDI_FTYPE_UDI_UDI, crypto_zbkx64),

__builtin_riscv_xperm8(rs1,rs2);
```

怎么添加

以riscv为例

```
RISCV_BUILTIN (xperm8_di, "xperm8", RISCV_BUILTIN_DIRECT,
RISCV_UDI_FTYPE_UDI_UDI, crypto_zbkx64),

__builtin_riscv_xperm8(rs1,rs2);
```

怎么添加

以riscv为例

riscv_bitmanip.h

```
#if defined(__riscv_zbkx) && __riscv_xlen == 64
extern __inline uint64_t
__attribute__ ((__gnu_inline__, __always_inline__, __artificial__))
__riscv_xperm8_64 (uint64_t rs1, uint64_t rs2)
{
    return __builtin_riscv_xperm8 (rs1,rs2);
}
```

谢 谢

欢迎交流合作

2023/12/15