Porting ZFINX Support on RISC-V GNU Toolchain

PLCT实验室 陈嘉炜

2020.11.20

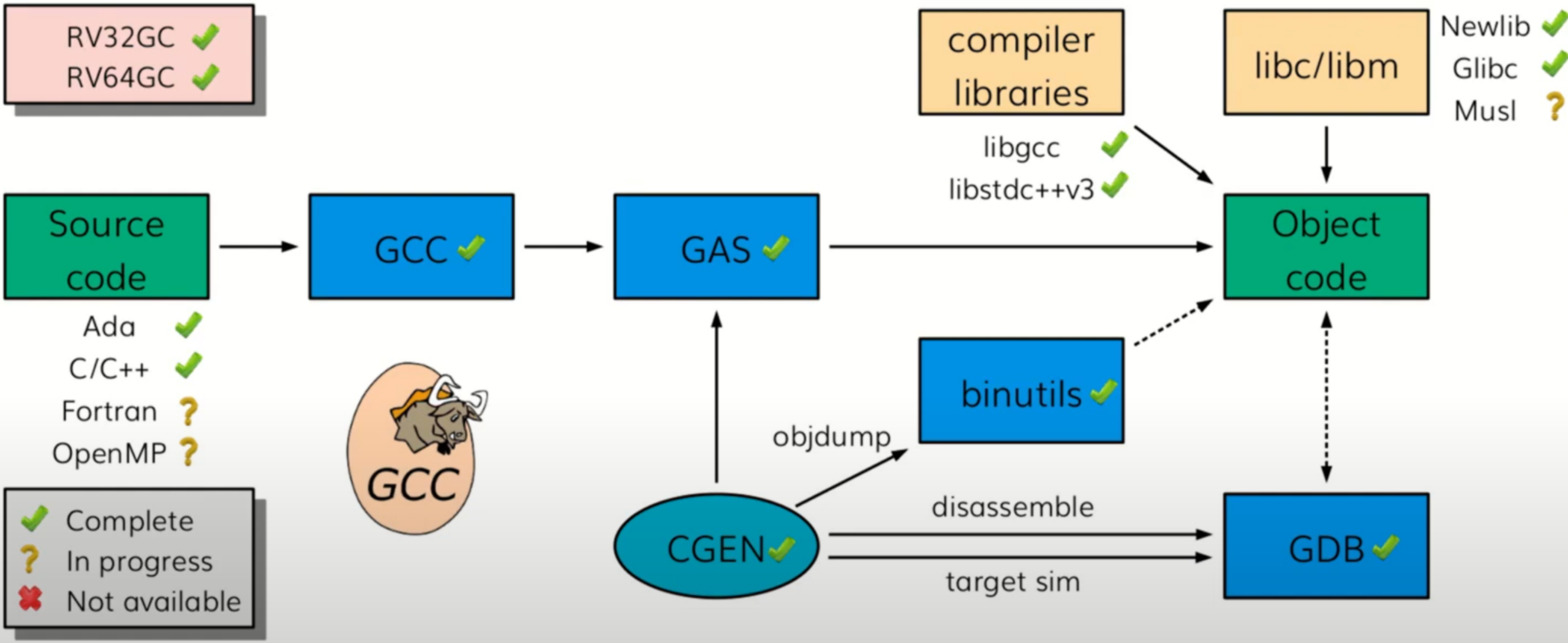
项目地址：

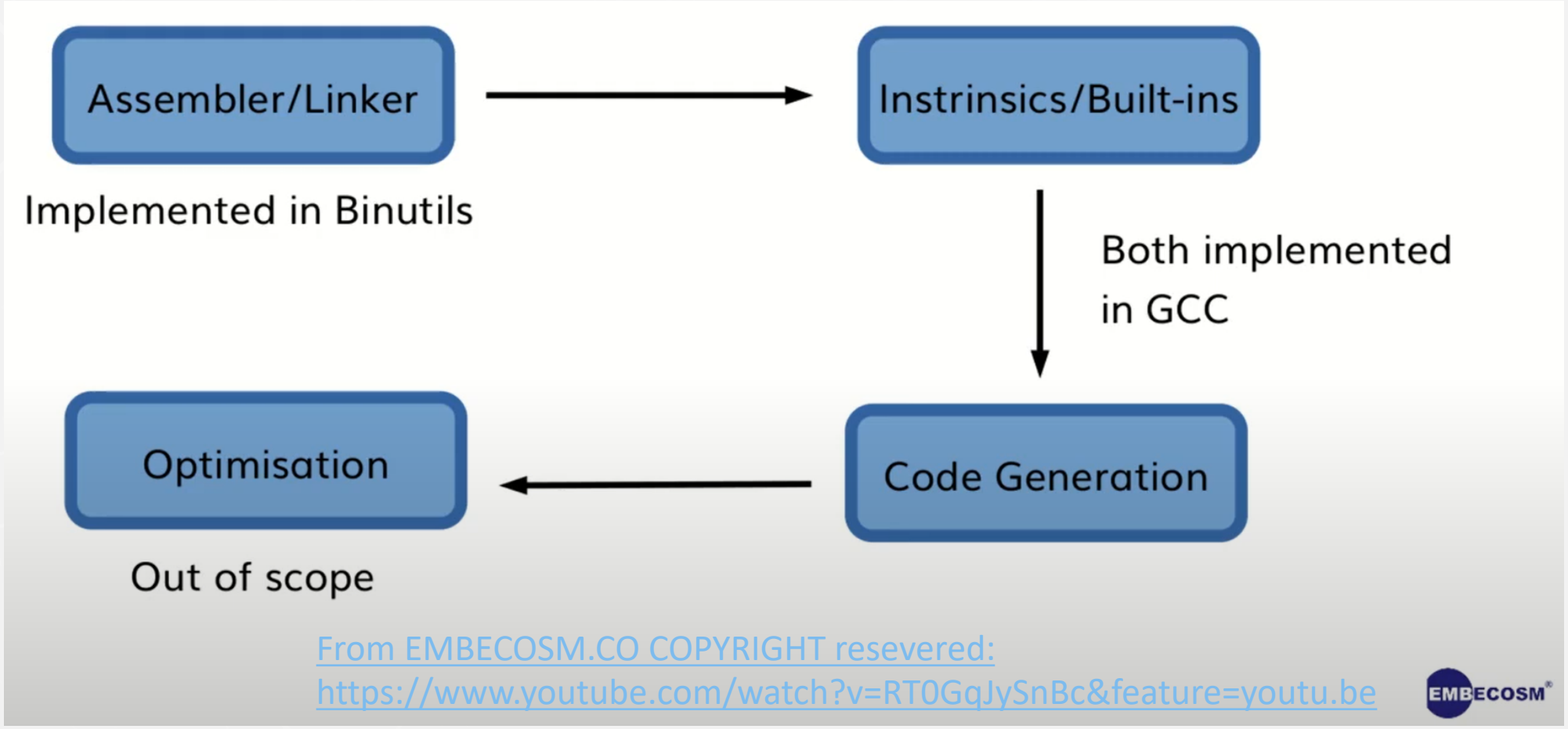
https://github.com/pz9115/riscv-binutils-gdb/tree/riscv-binutils-2.35-zfinx

https://github.com/pz9115/riscv-gcc/tree/riscv-gcc-10.2.0-zfinx

持续跟新中

RISC-V GNU Tool Chain Components




Assembler/Linker
Instrinsics/Built-ins
Implemented in Binutils
Both implemented
in GCC
Optimisation
Code Generation
Out of scope
From EMBECOSM.CO COPYRIGHT resevered:
https://www.youtube.com/watch?v=RT0GqJySnBc&feature=youtu.be
EMBECOSM®

RISC-V GCC简介：

riscv-gcc/gcc/common/config/riscv/riscv-common.c [commit c9cb50e]

```
/* Mapping table between extension to internal flag.  */
static const riscv_ext_flag_table_t riscv_ext_flag_table[] =
{
  {"e", &gcc_options::x_target_flags, MASK_RVE},
  {"m", &gcc_options::x_target_flags, MASK_MUL},
  {"a", &gcc_options::x_target_flags, MASK_ATOMIC},
  {"f", &gcc_options::x_target_flags, MASK_HARD_FLOAT},
  {"d", &gcc_options::x_target_flags, MASK_DOUBLE_FLOAT},
  {"c", &gcc_options::x_target_flags, MASK_RVC},
  {"zfinx", &gcc_options::x_target_flags, MASK_ZFINX},
  {NULL, NULL, 0}
};
```

RISC-V GCC文件目录：

主要修改的文件有—

constraints.md（条件约束）

riscv.c（ABI 定义）

riscv.h （寄存器、参数定义）

riscv.md （指令模板）

```
root@plct:~/riscv-gnu-toolchain/riscv-gcc/gcc/config/riscv# tree -s
.
├── [       2663]  constraints.md
├── [       1171]  elf.h
├── [       1974]  freebsd.h
├── [       2742]  generic.md
├── [       2317]  linux.h
├── [       6797]  multilib-generator
├── [       1620]  peephole.md
├── [       4860]  pic.md
├── [       6734]  predicates.md
├── [       9384]  riscv-builtins.c
├── [     160700]  riscv.c
├── [       2647]  riscv-c.c
├── [       2094]  riscv-cores.def
├── [       1179]  riscv-d.c
├── [       1155]  riscv-ftypes.def
├── [      34994]  riscv.h
├── [      76327]  riscv.md
├── [        861]  riscv-modes.def
├── [       4358]  riscv.opt
├── [       1417]  riscv-opts.h
├── [        835]  riscv-passes.def
├── [       4386]  riscv-protos.h
├── [       5117]  riscv-shorten-memrefs.c
├── [      15235]  riscv-sr.c
├── [       1269]  rtems.h
├── [       3794]  sifive-7.md
├── [       6751]  sync.md
├── [       1119]  t-elf-multilib
├── [        232]  t-linux
├── [       2014]  t-linux-multilib
├── [       1094]  t-riscv
├── [       1188]  t-rtems
├── [        465]  t-withmultilib
├── [         98]  t-withmultilib-generator
└── [       1846]  withmultilib.h
```

constraints.md　　寄存器类型、XLEN(位长)约束

```
;; Zfinx support need refuse FP regs

(define_register_constraint "f"
  "TARGET_HARD_FLOAT ? (TARGET_ZFINX ? GR_REGS : FP_REGS) : NO_REGS"
  "A floating-point register (if available).")

;; General constraints

(define_constraint "I"
  "An I-type 12-bit signed immediate."
  (and (match_code "const_int")
       (match_test "SMALL_OPERAND (ival)")))

(define_constraint "J"
  "Integer zero."
  (and (match_code "const_int")
       (match_test "ival == 0")))

(define_constraint "K"
  "A 5-bit unsigned immediate for CSR access instructions."
  (and (match_code "const_int")
       (match_test "IN_RANGE (ival, 0, 31)")))

(define_constraint "L"
  "A U-type 20-bit signed immediate."
  (and (match_code "const_int")
       (match_test "LUI_OPERAND (ival)")))
```

riscv.c　　ABI定义、参数符

```
  /* This is the new ABI, which is the same for C++ and C.  */
  unsigned num_int_new = 0, num_float_new = 0;
  int n_new = riscv_flatten_aggregate_argument (type, fields, true);
  for (int i = 0; i < n_new; i++)
    {
      num_float_new += SCALAR_FLOAT_TYPE_P (fields[i].type);
      num_int_new += INTEGRAL_TYPE_P (fields[i].type);
    }

/* Require that the ISA supports the requested floating-point ABI.  */
if (UNITS_PER_FP_ARG > (TARGET_HARD_FLOAT ? UNITS_PER_FP_REG : 0))
  error ("requested ABI requires %<-march%> to subsume the %qc extension",
         UNITS_PER_FP_ARG > 8 ? 'Q' : (UNITS_PER_FP_ARG > 4 ? 'D' : 'F'));

if (TARGET_RVE && riscv_abi != ABI_ILP32E)
  error ("rv32e requires ilp32e ABI");

// Zfinx only supports floating-point arguments in X-registers.
//if (TARGET_ZFINX && riscv_abi != ABI_ILP32 && riscv_abi != ABI_LP64 && riscv_abi != ABI_ILP32E)
//  error ("zfinx requires ilp32e ABI or ilp32, lp64");
```

riscv.h

寄存器表与值

```
#define REGISTER_NAMES                                                  \
{ "zero","ra",  "sp",  "gp",  "tp",  "t0",  "t1",  "t2",            \
  "s0",  "s1",  "a0",  "a1",  "a2",  "a3",  "a4",  "a5",            \
  "a6",  "a7",  "s2",  "s3",  "s4",  "s5",  "s6",  "s7",            \
  "s8",  "s9",  "s10", "s11", "t3",  "t4",  "t5",  "t6",            \
  "ft0", "ft1", "ft2", "ft3", "ft4", "ft5", "ft6", "ft7",           \
  "fs0", "fs1", "fa0", "fa1", "fa2", "fa3", "fa4", "fa5",           \
  "fa6", "fa7", "fs2", "fs3", "fs4", "fs5", "fs6", "fs7",           \
  "fs8", "fs9", "fs10","fs11","ft8", "ft9", "ft10","ft11",          \
  "arg", "frame", }
```

XLEN、ABI对应关系

```
#define XLEN_SPEC \
  "%{march=rv32*:32}" \
  "%{march=rv64*:64}" \

#define ABI_SPEC \
  "%{mabi=ilp32:ilp32}" \
  "%{mabi=ilp32e:ilp32e}" \
  "%{mabi=ilp32f:ilp32f}" \
  "%{mabi=ilp32d:ilp32d}" \
  "%{mabi=lp64:lp64}" \
  "%{mabi=lp64f:lp64f}" \
  "%{mabi=lp64d:lp64d}" \
```

riscv.h

OPCODE_MASK

```
/* ISA constants needed for code generation.  */
#define OPCODE_LW    0x2003
#define OPCODE_LD    0x3003
#define OPCODE_AUIPC 0x17
#define OPCODE_JALR  0x67
#define OPCODE_LUI   0x37
#define OPCODE_ADDI  0x13
#define SHIFT_RD  7
#define SHIFT_RS1 15
#define SHIFT_IMM 20
#define IMM_BITS 12
#define C_S_BITS 5
#define C_SxSP_BITS 6
```

riscv.md

RTL模板，
用于汇编指令生成

```
;;
;;  ....................
;;
;;      ADDITION
;;
;;  ....................
;;

(define_insn "add<mode>3"
  [(set (match_operand:ANYF           0 "register_operand" "=f")
        (plus:ANYF (match_operand:ANYF 1 "register_operand" " f")
                   (match_operand:ANYF 2 "register_operand" " f")))]
  "TARGET_HARD_FLOAT"
  "fadd.<fmt>\t%0,%1,%2"
  [(set_attr "type" "fadd")
   (set_attr "mode" "<UNITMODE>")])

(define_insn "addsi3"
  [(set (match_operand:SI          0 "register_operand" "=r,r")
        (plus:SI (match_operand:SI 1 "register_operand" " r,r")
                 (match_operand:SI 2 "arith_operand"    " r,I")))]
  ""
  { return TARGET_64BIT ? "add%i2w\t%0,%1,%2" : "add%i2\t%0,%1,%2"; }
  [(set_attr "type" "arith")
   (set_attr "mode" "SI")])
```

RISC-V BINUTILS简介：

要修改BINUTILS支持RISC-V-ZFINX扩展指令

```
root@plct:~/test# /opt/riscv/bin/riscv64-unknown-elf-readelf fo.elf -h
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           RISC-V
  Version:                           0x1
  Entry point address:               0x100c2
  Start of program headers:          64 (bytes into file)
  Start of section headers:          130224 (bytes into file)
  Flags:                             0x5, RVC, zfinx ABI
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         2
  Size of section headers:           64 (bytes)
  Number of section headers:         23
  Section header string table index: 22
```

ELF header添加：

/riscv-binutils/bfd/elfnn-riscv.c

/riscv-binutils/bfd/elfxx-riscv.c

commit db46cc8

```
riscv_float_abi_string (flagword flags)
{
  switch (flags & EF_RISCV_FLOAT_ABI)
    {
    case EF_RISCV_FLOAT_ABI_SOFT:
      return "soft-float";
      break;
    case EF_RISCV_FLOAT_ABI_SINGLE:
      return "single-float";
      break;
    case EF_RISCV_FLOAT_ABI_DOUBLE:
      return "double-float";
      break;
    case EF_RISCV_FLOAT_ABI_QUAD:
      return "quad-float";
      break;
    case EF_RISCV_ZFINX_ABI:
      return "zfinx";
      break;
    default:
      abort ();
    }
}
```

```
/* First letter must start with i, e or g.  */
switch (*p)
  {
    case 'i':
      p = riscv_parsing_subset_version (rps,
                                        march,
                                        ++p,
                                        &major_version,
                                        &minor_version,
                                        /* std_ext_p= */TRUE,
                                        &use_default_version);

      /* Find the default version if needed.  */
      if (use_default_version
          && rps->get_default_version != NULL)
        rps->get_default_version ("i",
                                  &major_version,
                                  &minor_version);
      riscv_add_subset (rps->subset_list, "i",
                        major_version, minor_version);
      break;

    case 'e':
      p = riscv_parsing_subset_version (rps,
                                        march,
                                        ++p,
                                        &major_version,
                                        &minor_version,
                                        /* std_ext_p= */TRUE,
                                        &use_default_version);
```

/riscv-binutils/include/elf/riscv.h　　ABI_CODE（待定）

```
/* Which floating-point ABI a file uses.  */
#define EF_RISCV_FLOAT_ABI 0x0006

/* File uses the soft-float ABI.  */
#define EF_RISCV_FLOAT_ABI_SOFT 0x0000

/* File uses the zfinx ABI. */
#define EF_RISCV_ZFINX_ABI 0x000A

/* File uses the single-float ABI.  */
#define EF_RISCV_FLOAT_ABI_SINGLE 0x0002

/* File uses the double-float ABI.  */
#define EF_RISCV_FLOAT_ABI_DOUBLE 0x0004

/* File uses the quad-float ABI.  */
#define EF_RISCV_FLOAT_ABI_QUAD 0x0006

/* File uses the 32E base integer instruction.  */
#define EF_RISCV_RVE 0x0008
```

/riscv-binutils/include/gas/riscv.h

```
/* All RISC-V instructions belong to at least one of these classes.  */

enum riscv_insn_class
  {
   INSN_CLASS_NONE,

   INSN_CLASS_I,
   INSN_CLASS_C,
   INSN_CLASS_A,
   INSN_CLASS_M,
   INSN_CLASS_F,
   INSN_CLASS_D,
   INSN_CLASS_ZFINX,
   INSN_CLASS_D_AND_C,
   INSN_CLASS_F_AND_C,
   INSN_CLASS_ZFINX_AND_C,
   INSN_CLASS_Q,
  };
```

/riscv-binutils/gas/config/tc-riscv.c

```
{"fadd.s",      0, INSN_CLASS_F,    "D,S,T",   MATCH_FADD_S | MASK_RM, MASK_FADD_S | MASK_RM, match_opcode, 0 },
{"fadd.s",      0, INSN_CLASS_F,    "D,S,T,m",  MATCH_FADD_S, MASK_FADD_S, match_opcode, 0 },
{"fadd.s",      0, INSN_CLASS_ZFINX,    "d,s,t",   MATCH_FADD_S | MASK_RM, MASK_FADD_S | MASK_RM, match_opcode, 0 },
{"fadd.s",      0, INSN_CLASS_ZFINX,    "d,s,t,m",  MATCH_FADD_S, MASK_FADD_S, match_opcode, 0 },
```

```
case 'S': /* Floating-point RS1 x8-x15.  */
  if (!reg_lookup (&s, RCLASS_FPR, &regno)
      || !(regno >= 8 && regno <= 15))
    break;
  INSERT_OPERAND (CRS1S, *ip, regno % 8);
  continue;
case 'D': /* Floating-point RS2 x8-x15.  */
  if (!reg_lookup (&s, RCLASS_FPR, &regno)
      || !(regno >= 8 && regno <= 15))
    break;
  INSERT_OPERAND (CRS2S, *ip, regno % 8);
  continue;
case 'T': /* Floating-point RS2.  */
  if (!reg_lookup (&s, RCLASS_FPR, &regno))
    break;
  INSERT_OPERAND (CRS2, *ip, regno);
  continue;
```

```
case 'd':           /* Destination register.  */
case 's':           /* Source register.  */
case 't':           /* Target register.  */
case 'r':           /* rs3.  */
```

```
case 's':
  INSERT_OPERAND (RS1, *ip, regno);
  break;
case 'd':
  INSERT_OPERAND (RD, *ip, regno);
  break;
case 't':
  INSERT_OPERAND (RS2, *ip, regno);
  break;
case 'r':
  INSERT_OPERAND (RS3, *ip, regno);
  break;
```

/riscv-binutils/opcodes/riscv-opc.c （编码）
/riscv-binutils/opcodes/riscv-dis.c （解码）

```
{"fadd.s",      0, INSN_CLASS_F,    "D,S,T",   MATCH_FADD_S | MASK_RM, MASK_FADD_S | MASK_RM, match_opcode, 0 },
{"fadd.s",      0, INSN_CLASS_F,    "D,S,T,m",   MATCH_FADD_S, MASK_FADD_S, match_opcode, 0 },
{"fadd.s",      0, INSN_CLASS_ZFINX,    "d,s,t",   MATCH_FADD_S | MASK_RM, MASK_FADD_S | MASK_RM, match_opcode, 0 },
{"fadd.s",      0, INSN_CLASS_ZFINX,    "d,s,t,m",   MATCH_FADD_S, MASK_FADD_S, match_opcode, 0 },
{"fsub.s",      0, INSN_CLASS_F,    "D,S,T",   MATCH_FSUB_S | MASK_RM, MASK_FSUB_S | MASK_RM, match_opcode, 0 },
{"fsub.s",      0, INSN_CLASS_F,    "D,S,T,m",   MATCH_FSUB_S, MASK_FSUB_S, match_opcode, 0 },
{"fsub.s",      0, INSN_CLASS_ZFINX,    "d,s,t",   MATCH_FSUB_S | MASK_RM, MASK_FSUB_S | MASK_RM, match_opcode, 0 },
{"fsub.s",      0, INSN_CLASS_ZFINX,    "d,s,t,m",   MATCH_FSUB_S, MASK_FSUB_S, match_opcode, 0 },
```

OP_NAME, XLEN, INSN_CLASS, OP_FUN,OP_MASK, MATCH_FUN, PINFO

谢谢