从零开始的RISC-V模拟器开发

第8讲 QEMU篇之CPU虚拟化2

中国科学院软件研究所
PLCT实验室

王俊强 wangjunqiang@iscas.ac.cn
李威威 liweiwei@iscas.ac.cn
吴伟 wuwei2016@iscas.ac.cn

本课内容

CPU虚拟化

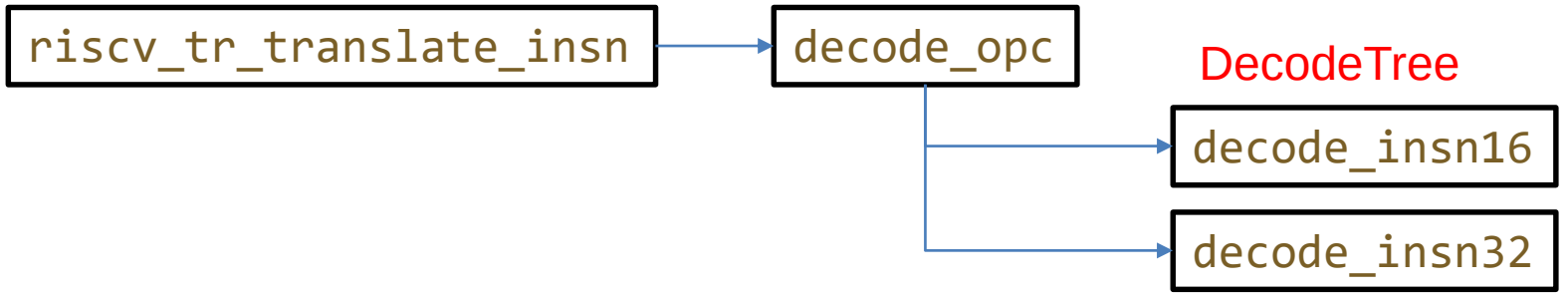
- ~~RISCV CPU实现分析~~
- ~~新RISCV CPU建立~~
- ~~TCG原理与指令翻译~~
- DecodeTree与指令添加
- CSR寄存器实现添加

指令添加DecodeTree

target/arch/translate.c

```
struct TranslatorOps
    void (*init_disas_context)
    void (*tb_start)
    void (*insn_start)
    bool (*breakpoint_check)
    void (*translate_insn)
    void (*tb_stop)
    void (*disas_log)
```

```
static void riscv_tr_translate_insn(DisasContextBase *dcbase, CPUState
 *cpu)
{
    ……
    uint16_t opcode16 = translator_lduw(env, ctx->base.pc_next);

    decode_opc(env, ctx, opcode16);
    ctx->base.pc_next = ctx->pc_succ_insn;

    ……
}
```



文件：target/riscv/meson.build

```
gen32 = [
  decodetree.process('insn16.decode', extra_args: [dir / 'insn16-32.decode', '--static-decode=decode_insn16', '--insnwidth=16']),
  decodetree.process('insn32.decode', extra_args: '--static-decode=decode_insn32'),
]
```

指令添加DecodeTree

meson.build

```
if have_system or have_user
  decodetree = generator(find_program('scripts/decodetree.py'),
                         output: 'decode-@BASENAME@.c.inc',
                         arguments: ['@INPUT@', '@EXTRA_ARGS@', '-o', '@OUTPUT@'])
  subdir('libdecnumber')
  subdir('target')
endif
```

选项：
--translate：指定translator function 默认trans_
--decode：设置decode function 默认decode
--static-decode：设置static decode function
-o / --output：生成输出文件
-w / --insnwidth：固定指令长度
--varinsnwidth：变化指令长度
默认输入文件

```
gen32 = [
  decodetree.process('insn16.decode', extra_args: [dir / 'insn16-32.decode', '--static-decode=decode_insn16', '--insnwidth=16']),
  decodetree.process('insn32.decode', extra_args: '--static-decode=decode_insn32'),
]

gen64 = [
  decodetree.process('insn16.decode', extra_args: [dir / 'insn16-64.decode', '--static-decode=decode_insn16', '--insnwidth=16']),
  decodetree.process('insn32.decode', extra_args: [dir / 'insn32-64.decode', '--static-decode=decode_insn32']),
]
```

https://mesonbuild.com/Generating-sources.html#using-generator meson: https://mesonbuild.com/

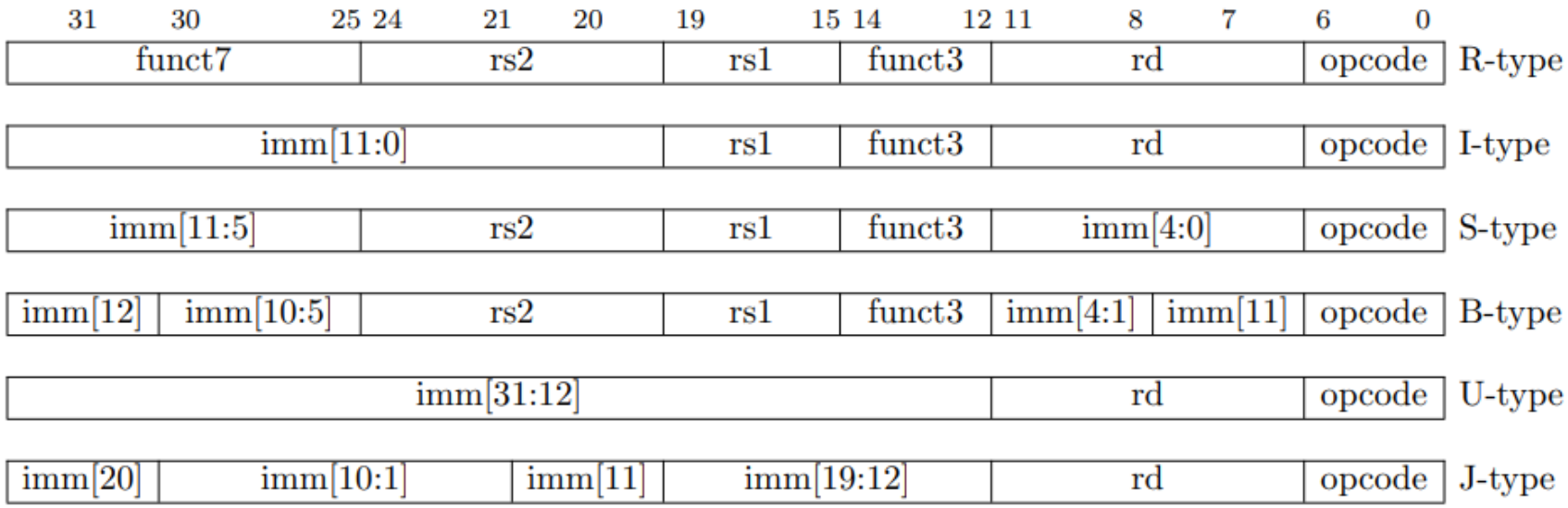
DecodeTree介绍

Decodetree 语法组成:

Fields、Argument Sets、Formats、Patterns、Pattern Groups

取值范围　参数结构体　取值方法　单指令描述　多指令组合

rs2 rs1 rd　u or s　16bit or 32bit　同opcode

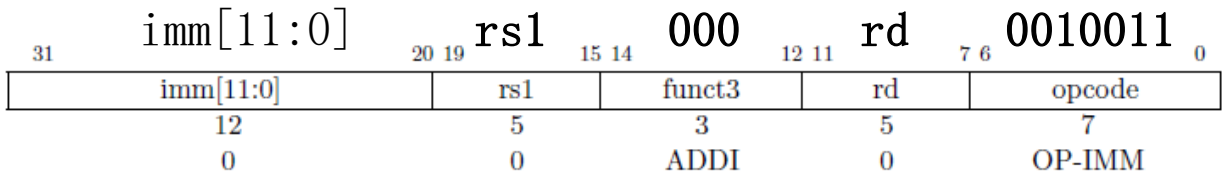


RISC-V base instruction formats showing immediate variants.来自:riscv-v spec

https://qemu.readthedocs.io/en/latest/devel/decodetree.html

DecodeTree之Fields

NOP Instruction



ADDI adds the sign-extended 12-bit immediate to register rs1.

语法:

```
field_def     := '%' identifier ( unnamed_field )* ( !function=identifier )?
unnamed_field := number ':' ( 's' ) number
```

addi rd, rs1, immediate
x[rd] = x[rs1] + sext(immediate)

Field examples:

Input	Generated code
%disp 0:s16	sextract(i, 0, 16)
%imm9 16:6 10:3	extract(i, 16, 6) << 3 | extract(i, 10, 3)
%disp12 0:s1 1:1 2:10	sextract(i, 0, 1) << 11 |extract(i, 1, 1) << 10 | extract(i, 2, 10)
%shimm8 5:s8 13:1!function=expand_shimm8	expand_shimm8(sextract(i, 5, 8) << 1 |extract(i, 13, 1))

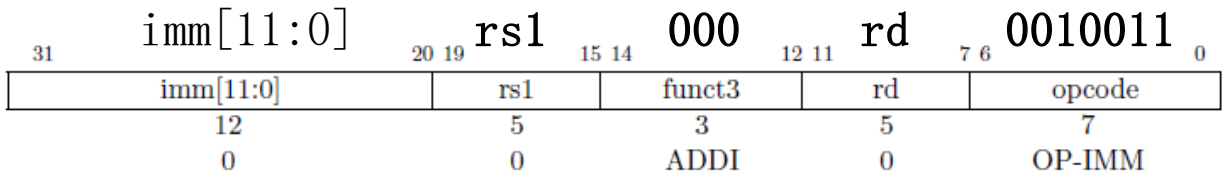
```
# Fields:
%rs1       15:5
%rd        7:5

# immediates:
%imm_i    20:s12
```

```
static void decode_insn32_extract_i(DisasContext *ctx, arg_i *a, uint32_t insn)
{
    a->imm = sextract32(insn, 20, 12);
    a->rs1 = extract32(insn, 15, 5);
    a->rd = extract32(insn, 7, 5);
}
```

https://qemu.readthedocs.io/en/latest/devel/decodetree.html

DecodeTree之Argument Sets

NOP Instruction



imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
0	0	ADDI	0	OP-IMM

ADDI adds the sign-extended 12-bit immediate to register rs1.

语法:

```
args_def    := '&' identifier ( args_elt )+ ( !extern )?
args_elt    := identifier (':' identifier)?
```

addi rd, rs1, immediate
x[rd] = x[rs1] + sext(immediate)

Argument set examples:

```
&reg3       ra rb rc
&loadstore  reg base offset
&longldst   reg base offset:int64_t
```

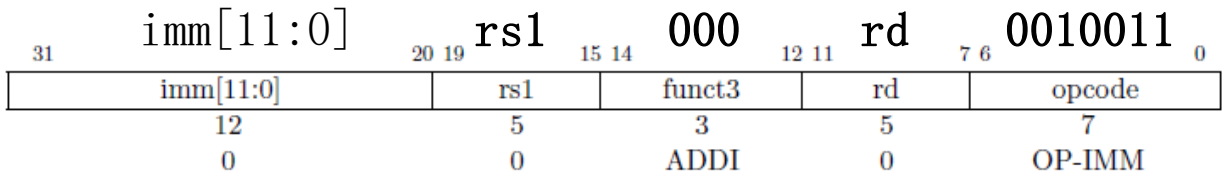
```
# Argument sets:
&i    imm rs1 rd
```

```
typedef struct {
    int imm;
    int rd;
    int rs1;
} arg_i;

typedef arg_i arg_addi;
```

https://qemu.readthedocs.io/en/latest/devel/decodetree.html

DecodeTree之Formats

NOP Instruction



imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
0	0	ADDI	0	OP-IMM

ADDI adds the sign-extended 12-bit immediate to register rs1.

addi rd, rs1, immediate
x[rd] = x[rs1] + sext(immediate)

语法:

```
fmt_def      := '@' identifier ( fmt_elt )+
fmt_elt      := fixedbit_elt | field_elt | field_ref | args_ref
fixedbit_elt := [01.-]+
field_elt    := identifier ':' 's'? number
field_ref    := '%' identifier | identifier '=' '%' identifier
args_ref     := '&' identifier
```

Format examples:

```
@opr    ...... ra:5 rb:5 ... 0 ....... rc:5
@opi    ...... ra:5 lit:8    1 ....... rc:5
```

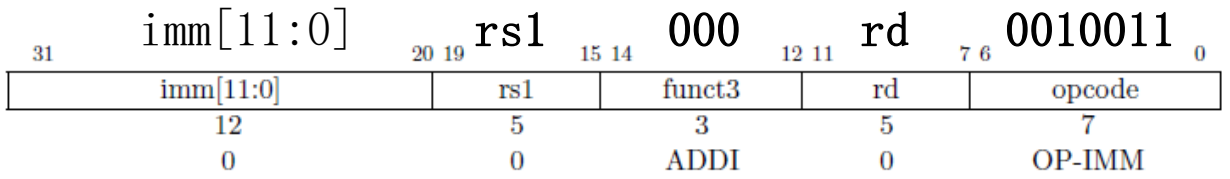
```
# Formats 32:
@i       ...........     ..... ... ..... ....... &i      imm=%imm_i     %rs1 %rd
```

```
static void decode_insn32_extract_i(D
isasContext *ctx, arg_i *a, uint32_t
insn)
{
    a->imm = sextract32(insn, 20, 12);
    a->rs1 = extract32(insn, 15, 5);
    a->rd = extract32(insn, 7, 5);
}
```

https://qemu.readthedocs.io/en/latest/devel/decodetree.html

DecodeTree之Patterns

NOP Instruction



imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
0	0	ADDI	0	OP-IMM

ADDI adds the sign-extended 12-bit immediate to register rs1.

语法:

```
pat_def     := identifier ( pat_elt )+
pat_elt     := fixedbit_elt | field_elt | field_ref | args_ref | fmt_ref | const_elt
fmt_ref     := '@' identifier
const_elt   := identifier '=' number
```

addi rd, rs1, immediate
x[rd] = x[rs1] + sext(immediate)

```
# *** RV32I Base Instruction Set ***
addi     ...........     ..... 000 ..... 0010011 @i
```

Pattern examples:

```
addl_r   010000 ..... ..... .... 0000000 ..... @opr
addl_i   010000 ..... ..... .... 0000000 ..... @opi
```

```
trans_addl_r(ctx, &arg_opr, insn)
```

```
trans_addl_i(ctx, &arg_opi, insn)
```

```
static bool trans_addi(DisasContext *ctx, arg_addi *a);
```

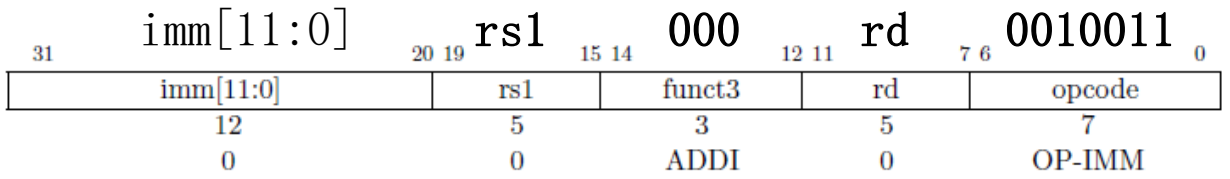
target/riscv/decode_insn32.inc.c

```
static bool decode_insn32(DisasContext *ctx, uint32_t insn)
{
    switch (insn & 0x0000007f) {
        ……
        case 0x00000013:
            /* ........ ........ ........ .0010011 */
            switch ((insn >> 12) & 0x7) {
            case 0x0:
                /* ........ ........ .000.... .0010011 */
                decode_insn32_extract_i(ctx, &u.f_i, insn);
                if (trans_addi(ctx, &u.f_i)) return true;
                return false;
                ……
            }
        ……
    }
}
```

https://qemu.readthedocs.io/en/latest/devel/decodetree.html

DecodeTree之Patterns

NOP Instruction



ADDI adds the sign-extended 12-bit immediate to register rs1.

语法:

```
pat_def     := identifier ( pat_elt )+
pat_elt     := fixedbit_elt | field_elt | field_ref | args_ref | fmt_ref | const_elt
fmt_ref     := '@' identifier
const_elt   := identifier '=' number
```

Pattern examples:

```
addl_r   010000 ..... ..... .... 0000000 ..... @opr
addl_i   010000 ..... ..... .... 0000000 ..... @opi
```

```
trans_addl_r(ctx, &arg_opr, insn)
```

```
trans_addl_i(ctx, &arg_opi, insn)
```

target/riscv/insn_trans/trans_rvi.inc.c

```c
static bool trans_addi(DisasContext *ctx, arg_addi *a)
{
    return gen_arith_imm_fn(ctx, a, &tcg_gen_addi_tl);
}

#define tcg_gen_addi_tl tcg_gen_addi_i32

void tcg_gen_addi_i32(TCGv_i32 ret, TCGv_i32 arg1, int32_t arg2)
{
    /* some cases can be optimized here */
    if (arg2 == 0) {
        tcg_gen_mov_i32(ret, arg1);
    } else {
        TCGv_i32 t0 = tcg_const_i32(arg2);
        tcg_gen_add_i32(ret, arg1, t0);
        tcg_temp_free_i32(t0);
    }
}
```

参考:
Backend Ops
Frontend Ops

https://qemu.readthedocs.io/en/latest/devel/decodetree.html

DecodeTree之Pattern Groups

语法:

```
group           := overlap_group | no_overlap_group
overlap_group   := '{' ( pat_def | group )+ '}'
no_overlap_group := '[' ( pat_def | group )+ ']'
```

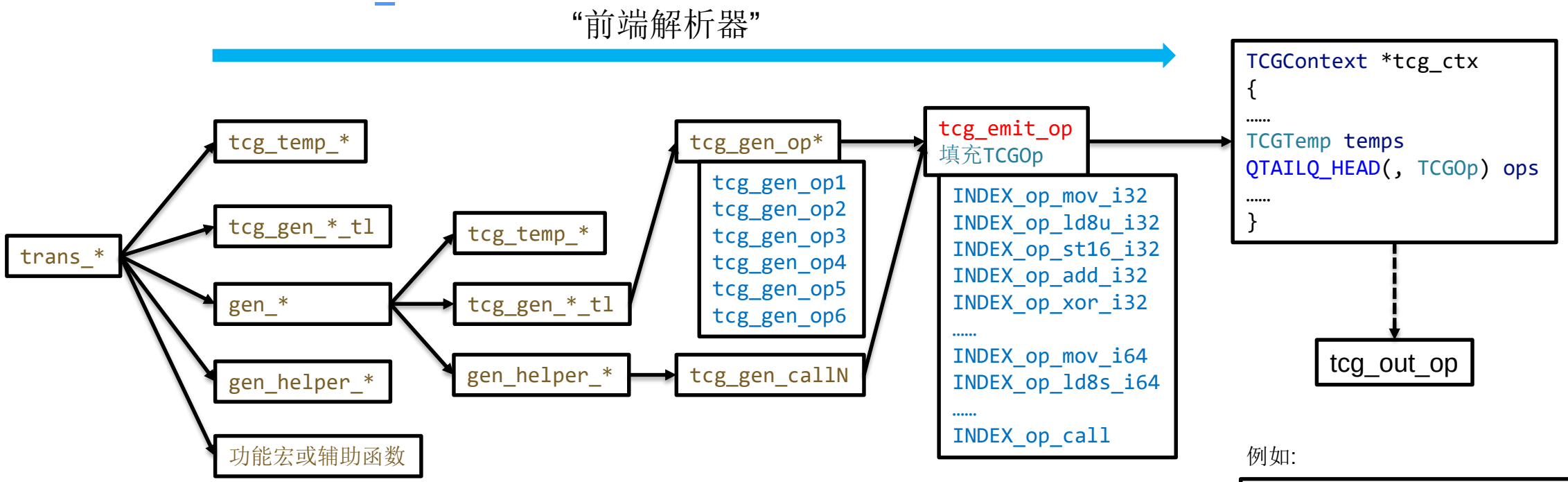
Pattern Groups examples:

```
{
 {
  nop   000010 ----- ----- 0000 001001 0 00000
  copy  000010 00000 r1:5  0000 001001 0 rt:5
 }
 or     000010 rt2:5 r1:5  cf:4 001001 0 rt:5
}
```

```
switch (insn & 0xfc000fe0) {
case 0x08000240:
  /* 000010.. ........ ....0010 010..... */
  if ((insn & 0x0000f000) == 0x00000000) {
      /* 000010.. ........ 00000010 010..... */
      if ((insn & 0x0000001f) == 0x00000000) {
          /* 000010.. ........ 00000010 01000000 */
          extract_decode_Fmt_0(&u.f_decode0, insn);
          if (trans_nop(ctx, &u.f_decode0)) return true;
      }
      if ((insn & 0x03e00000) == 0x00000000) {
          /* 00001000 000..... 00000010 010..... */
          extract_decode_Fmt_1(&u.f_decode1, insn);
          if (trans_copy(ctx, &u.f_decode1)) return true;
      }
  }
  extract_decode_Fmt_2(&u.f_decode2, insn);
  if (trans_or(ctx, &u.f_decode2)) return true;
  return false;
}
```

https://qemu.readthedocs.io/en/latest/devel/decodetree.html

指令添加之trans_*函数



例如:

```
#define REQUIRE_EXT(ctx, ext) do { \
    if (!has_ext(ctx, ext)) {      \
        return false;              \
    }                              \
} while (0)

static inline bool has_ext(DisasCont
ext *ctx, uint32_t ext)
{
    return ctx->misa & ext;
}
```

tl => target_long

```
typedef int32_t target_long;
typedef uint32_t target_ulong;
or
typedef int64_t target_long;
typedef uint64_t target_ulong;
```

例如:

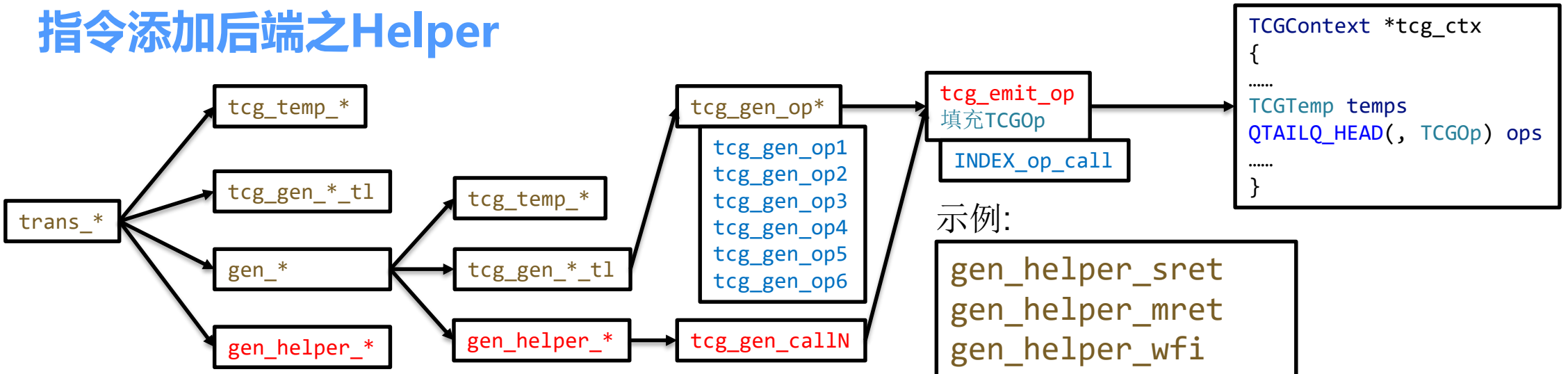
```
trans_addi
>>gen_arith_imm_fn
  >>gen_get_gpr
    >>tcg_gen_movi_tl
    >>tcg_gen_mov_tl
        >>tcg_gen_mov_i32
          >>tcg_gen_op2_i32
            >>tcg_gen_op2
              >>tcg_emit_op
```

指令添加之前端

前端类别	示例	含义
Registers	TCGv reg = tcg_global_mem_new(TCG_AREG0, offsetof(CPUState, reg), "reg");	Declare a named TCG register
Temporaries	TCGv tmp = tcg_temp_new();	Create a new temporary register
Labels	int l = gen_new_label();	Create a new label.
Math	tcg_gen_neg_tl(ret, arg1);	ret = arg1
Logic	tcg_gen_and_tl(ret, arg1, arg2);	ret = arg1 & arg2
Shift	tcg_gen_shl_tl(ret, arg1, arg2);	ret = arg1 << arg2
Rotation	tcg_gen_rotr_tl(ret, arg1, arg2);	ret = arg1 rotr arg2
Byte	tcg_gen_ext16u_tl(ret, arg1)	ret = (uint16_t)arg1
Load/Store	tcg_gen_ld64_tl(reg, cpu_env, offsetof(CPUState, reg));	Load a 64bit quantity from host memory
Code Flow	tcg_gen_brcond_tl(TCG_COND_XXX, arg1, arg2, label);	if (arg1 <condition> arg2) goto label

https://wiki.qemu.org/Documentation/TCG/frontend-ops

指令添加后端之Helper



gen_helper_? helper_ ?

示例:

```
gen_helper_sret
gen_helper_mret
gen_helper_wfi
gen_helper_fmadd_d
gen_helper_fadd_d
……
```

例子：给addi添加helper函数

1.在target/riscv/op_helper.c中定义

```
void helper_foo(CPURISCVState *env, target_ulong pc)
{
#ifdef  TARGET_RISCV32
    printf("hello addi in PC %x\n",  pc);
#else
    printf("hello addi in PC %lx\n", pc);
#endif
}
```

2.在target/riscv/helper.h中声明

```
DEF_HELPER_2(foo, void, env, tl)
```

3.在target/riscv/insn_trans/trans_rvi.c.inc中调用

```
static bool trans_addi(DisasContext *ctx, arg_addi *a)
{
    gen_helper_foo(cpu_env,  cpu_pc);
    return gen_arith_imm_fn(ctx, a, &tcg_gen_addi_tl);
}
```

指令添加后端之Helper

hello.s => hello

```
.global _start
_start: addi  a0, x0, 1
        la    a1, helloworld
        addi  a2, x0, 13
        addi  a7, x0, 64
        ecall
        addi    a0, x0, 0
        addi    a7, x0, 93
        ecall
.data
helloworld:  .ascii "Hello World\n"
```

运行：qemu-riscv32 hello

```
hello addi in PC 10074
hello addi in PC 10074
hello addi in PC 10074
hello addi in PC 10074
Hello World
hello addi in PC 1008c
hello addi in PC 1008c
```

$qemu-riscv32 -d in_asm hello

```
----------------
IN:
0x00010074:  00100513     addi     a0,zero,1
0x00010078:  00001597     auipc    a1,4096          # 0x11078
0x0001007c:  02058593     addi     a1,a1,32
0x00010080:  00d00613     addi     a2,zero,13
0x00010084:  04000893     addi     a7,zero,64
0x00010088:  00000073     ecall

Hello World!
----------------
IN:
0x0001008c:  00000513     mv       a0,zero
0x00010090:  05d00893     addi     a7,zero,93
0x00010094:  00000073     ecall
```

$qemu-riscv32 -d op_opt hello

```
---- 0001007c
call foo,$0x0,$0,env,pc                    dead: 0 1
add_i32 tmp0,x11/a1,$0x20                   dead: 1 2  pref=0xffff
mov_i32 x11/a1,tmp0                         sync: 0  dead: 0 1  pref=0xffff

---- 00010080
call foo,$0x0,$0,env,pc                    dead: 0 1
mov_i32 x12/a2,$0xd                         sync: 0  dead: 0 1  pref=0xffff

---- 00010084
call foo,$0x0,$0,env,pc                    dead: 0 1
mov_i32 x17/a7,$0x40                        sync: 0  dead: 0 1  pref=0xffff
```

指令添加后端之Helper

hello.s => hello

```
.global _start
_start: addi   a0, x0, 1
        la     a1, helloworld
        addi   a2, x0, 13
        addi   a7, x0, 64
        ecall
        addi     a0, x0, 0
        addi     a7, x0, 93
        ecall
.data
helloworld:  .ascii "Hello World\n"
```

运行：qemu-riscv32 hello

```
hello addi in PC 10074
hello addi in PC 10074
hello addi in PC 10074
hello addi in PC 10074
Hello World
hello addi in PC 1008c
hello addi in PC 1008c
```

$qemu-riscv32 -d out_asm hello

```
  -- guest addr 0x00010080
0x7fd1380000d9:  c7 45 30 0d 00 00 00      movl     $0xd, 0x30(%rbp)
  -- guest addr 0x00010084
0x7fd1380000e0:  c7 45 44 40 00 00 00      movl     $0x40, 0x44(%rbp)
```

$qemu-riscv32 -d out_asm hello

```
  -- guest addr 0x00010080
0x7f0bf0000100:  48 8b fd                  movq     %rbp, %rdi
0x7f0bf0000103:  8b b5 94 05 00 00         movl     0x594(%rbp), %esi
0x7f0bf0000109:  ff 15 49 00 00 00         callq    *0x49(%rip)
0x7f0bf000010f:  c7 45 30 0d 00 00 00      movl     $0xd, 0x30(%rbp)
  -- guest addr 0x00010084
0x7f0bf0000116:  48 8b fd                  movq     %rbp, %rdi
0x7f0bf0000119:  8b b5 94 05 00 00         movl     0x594(%rbp), %esi
0x7f0bf000011f:  ff 15 33 00 00 00         callq    *0x33(%rip)
0x7f0bf0000125:  c7 45 44 40 00 00 00      movl     $0x40, 0x44(%rbp)
```

指令添加后端之Helper

target/riscv/helper.h中Helper函数的“声明”

```
DEF_HELPER_2(foo, void, env, tl)
```

helper函数声明

定义于:include/exec/helper-proto.h

```
#define DEF_HELPER_FLAGS_2(name, flags, ret, t1, t2) \
dh_ctype(ret) HELPER(name) (dh_ctype(t1), dh_ctype(t2));
```

定义于:include/exec/helper-tcg.h

```
#define DEF_HELPER_FLAGS_0(NAME, FLAGS, ret) \
  { .func = HELPER(NAME), .name = str(NAME), \
    .flags = FLAGS | dh_callflag(ret), \
    .sizemask = dh_sizemask(ret, 0) },
```

定义于:include/exec/helper-gen.h

```
#define DEF_HELPER_FLAGS_2(name, flags, ret, t1, t2)                \
static inline void glue(gen_helper_, name)(dh_retvar_decl(ret)      \
    dh_arg_decl(t1, 1), dh_arg_decl(t2, 2))                         \
{                                                                   \
  TCGTemp *args[2] = { dh_arg(t1, 1), dh_arg(t2, 2) };              \
  tcg_gen_callN(HELPER(name), dh_retvar(ret), 2, args);             \
}
```

定义于:include/exec/helper-head.h

```
#define DEF_HELPER_0(name, ret) \
    DEF_HELPER_FLAGS_0(name, 0, ret)
#define DEF_HELPER_1(name, ret, t1) \
    DEF_HELPER_FLAGS_1(name, 0, ret, t1)
#define DEF_HELPER_2(name, ret, t1, t2) \
    DEF_HELPER_FLAGS_2(name, 0, ret, t1, t2)
#define DEF_HELPER_3(name, ret, t1, t2, t3) \
    DEF_HELPER_FLAGS_3(name, 0, ret, t1, t2, t3)
……
#define DEF_HELPER_7(name, ret, t1, t2, t3, t4, t5, t6, t7) \
    DEF_HELPER_FLAGS_7(name, 0, ret, t1, t2, t3, t4, t5, t6, t7)
```

```
#ifndef glue
#define xglue(x, y) x ## y
#define glue(x, y) xglue(x, y)
……
#endif
```

```
#define HELPER(name) glue(helper_, name)
#define dh_ctype(t) dh_ctype_##t
#define dh_ctype_i32 uint32_t
#define dh_ctype_i32 uint32_t
#define dh_ctype_s32 int32_t
#define dh_ctype_int int
#define dh_ctype_i64 uint64_t
#define dh_ctype_s64 int64_t
#define dh_ctype_f16 uint32_t
#define dh_ctype_f32 float32
#define dh_ctype_f64 float64
#define dh_ctype_ptr void *
#define dh_ctype_cptr const void *
#define dh_ctype_void void
#define dh_ctype_noreturn void QEMU_NORETURN
```

指令添加后端之Helper

target/riscv/helper.h中Helper函数的“声明”

```
DEF_HELPER_2(foo, void, env, tl)
```

定义于:include/exec/helper-proto.h

```
#define DEF_HELPER_FLAGS_2(name, flags, ret, t1, t2) \
dh_ctype(ret) HELPER(name) (dh_ctype(t1), dh_ctype(t2));
```

定义于:include/exec/helper-tcg.h

```
#define DEF_HELPER_FLAGS_0(NAME, FLAGS, ret) \
  { .func = HELPER(NAME), .name = str(NAME), \
    .flags = FLAGS | dh_callflag(ret), \
    .sizemask = dh_sizemask(ret, 0) },
```

定义于:include/exec/helper-gen.h

```
#define DEF_HELPER_FLAGS_2(name, flags, ret, t1, t2)                \
static inline void glue(gen_helper_, name)(dh_retvar_decl(ret)      \
    dh_arg_decl(t1, 1), dh_arg_decl(t2, 2))                         \
{                                                                   \
  TCGTemp *args[2] = { dh_arg(t1, 1), dh_arg(t2, 2) };              \
  tcg_gen_callN(HELPER(name), dh_retvar(ret), 2, args);             \
}
```

定义于:include/exec/helper-head.h

```
#ifndef glue
#define xglue(x, y) x ## y
#define glue(x, y) xglue(x, y)
……
#endif
```

```
#define DEF_HELPER_0(name, ret) \
    DEF_HELPER_FLAGS_0(name, 0, ret)
#define DEF_HELPER_1(name, ret, t1) \
    DEF_HELPER_FLAGS_1(name, 0, ret, t1)
#define DEF_HELPER_2(name, ret, t1, t2) \
    DEF_HELPER_FLAGS_2(name, 0, ret, t1, t2)
#define DEF_HELPER_3(name, ret, t1, t2, t3) \
    DEF_HELPER_FLAGS_3(name, 0, ret, t1, t2, t3)
……
#define DEF_HELPER_7(name, ret, t1, t2, t3, t4, t5, t6, t7) \
    DEF_HELPER_FLAGS_7(name, 0, ret, t1, t2, t3, t4, t5, t6, t7)
```

helper函数注册

```
static const TCGHelperInfo all_helpers[] = {
#include "exec/helper-tcg.h"
};
static GHashTable *helper_table;//全局变量
```

tcg_context_init中

```
for (i = 0; i < ARRAY_SIZE(all_helpers); ++i) {
    g_hash_table_insert(helper_table,
                        (gpointer)all_helpers[i].func,
                        (gpointer)&all_helpers[i]);
}
```

指令添加后端之Helper

target/riscv/helper.h中Helper函数的“声明”

```
DEF_HELPER_2(foo, void, env, tl)
```

定义于:include/exec/helper-proto.h

```
#define DEF_HELPER_FLAGS_2(name, flags, ret, t1, t2) \
dh_ctype(ret) HELPER(name) (dh_ctype(t1), dh_ctype(t2));
```

定义于:include/exec/helper-tcg.h

```
#define DEF_HELPER_FLAGS_0(NAME, FLAGS, ret) \
  { .func = HELPER(NAME), .name = str(NAME), \
    .flags = FLAGS | dh_callflag(ret), \
    .sizemask = dh_sizemask(ret, 0) },
```

定义于:include/exec/helper-gen.h

```
#define DEF_HELPER_FLAGS_2(name, flags, ret, t1, t2)                \
static inline void glue(gen_helper_, name)(dh_retvar_decl(ret)      \
    dh_arg_decl(t1, 1), dh_arg_decl(t2, 2))                         \
{                                                                   \
  TCGTemp *args[2] = { dh_arg(t1, 1), dh_arg(t2, 2) };              \
  tcg_gen_callN(HELPER(name), dh_retvar(ret), 2, args);             \
}
```

gen_helper函数定义

定义于:include/exec/helper-head.h

```
#ifndef glue
#define xglue(x, y) x ## y
#define glue(x, y) xglue(x, y)
……
#endif
```

```
#define DEF_HELPER_0(name, ret) \
    DEF_HELPER_FLAGS_0(name, 0, ret)
#define DEF_HELPER_1(name, ret, t1) \
    DEF_HELPER_FLAGS_1(name, 0, ret, t1)
#define DEF_HELPER_2(name, ret, t1, t2) \
    DEF_HELPER_FLAGS_2(name, 0, ret, t1, t2)
#define DEF_HELPER_3(name, ret, t1, t2, t3) \
    DEF_HELPER_FLAGS_3(name, 0, ret, t1, t2, t3)
……
#define DEF_HELPER_7(name, ret, t1, t2, t3, t4, t5, t6, t7) \
    DEF_HELPER_FLAGS_7(name, 0, ret, t1, t2, t3, t4, t5, t6, t7)
```

```
#define dh_alias(t) glue(dh_alias_, t)
#define dh_retvar_decl(t) glue(dh_retvar_decl_, dh_alias(t))
#define dh_arg_decl(t, n) glue(TCGv_, dh_alias(t)) glue(arg, n)
#define dh_arg(t, n) \
  glue(glue(tcgv_, dh_alias(t)), _temp)(glue(arg, n))
#define dh_retvar(t) glue(dh_retvar_, dh_alias(t))
#define dh_alias_i32 i32
#define dh_alias_s32 i32
……
#define dh_retvar_decl_i32 TCGv_i32 retval,
#define dh_retvar_decl_i64 TCGv_i64 retval,
#define dh_retvar_decl_ptr TCGv_ptr retval,
……
#define dh_retvar_i32 tcgv_i32_temp(retval)
#define dh_retvar_i64 tcgv_i64_temp(retval)
#define dh_retvar_ptr tcgv_ptr_temp(retval)
……
```

指令添加后端之Helper

target/riscv/helper.h中Helper函数的“声明”

```
DEF_HELPER_2(foo, void, env, tl)
```

定义于:include/exec/helper-proto.h

```
#define DEF_HELPER_FLAGS_2(name, flags, ret, t1, t2) \
dh_ctype(ret) HELPER(name) (dh_ctype(t1), dh_ctype(t2));
```

定义于:include/exec/helper-tcg.h

```
#define DEF_HELPER_FLAGS_0(NAME, FLAGS, ret) \
  { .func = HELPER(NAME), .name = str(NAME), \
    .flags = FLAGS | dh_callflag(ret), \
    .sizemask = dh_sizemask(ret, 0) },
```

定义于:include/exec/helper-gen.h

```
#define DEF_HELPER_FLAGS_2(name, flags, ret, t1, t2)                \
static inline void glue(gen_helper_, name)(dh_retvar_decl(ret)      \
    dh_arg_decl(t1, 1), dh_arg_decl(t2, 2))                         \
{                                                                   \
  TCGTemp *args[2] = { dh_arg(t1, 1), dh_arg(t2, 2) };              \
  tcg_gen_callN(HELPER(name), dh_retvar(ret), 2, args);             \
}
```

定义于:include/exec/helper-head.h

```
#define DEF_HELPER_0(name, ret) \
    DEF_HELPER_FLAGS_0(name, 0, ret)
#define DEF_HELPER_1(name, ret, t1) \
    DEF_HELPER_FLAGS_1(name, 0, ret, t1)
#define DEF_HELPER_2(name, ret, t1, t2) \
    DEF_HELPER_FLAGS_2(name, 0, ret, t1, t2)
#define DEF_HELPER_3(name, ret, t1, t2, t3) \
    DEF_HELPER_FLAGS_3(name, 0, ret, t1, t2, t3)
……
#define DEF_HELPER_7(name, ret, t1, t2, t3, t4, t5, t6, t7) \
    DEF_HELPER_FLAGS_7(name, 0, ret, t1, t2, t3, t4, t5, t6, t7)
```

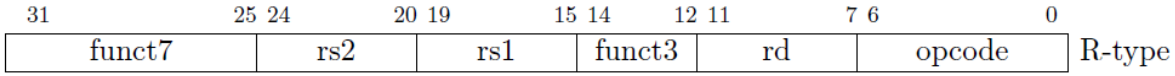
```
#ifndef glue
#define xglue(x, y) x ## y
#define glue(x, y) xglue(x, y)
……
#endif
```

```
void tcg_gen_callN(void *func, TCGTemp *ret, int nargs, TCGT
emp **args)
{
    int i, real_args, nb_rets, pi;
    unsigned sizemask, flags;
    TCGHelperInfo *info;
    TCGOp *op;

    info = g_hash_table_lookup(helper_table, (gpointer)func);
    flags = info->flags;
    sizemask = info->sizemask;
……
    op = tcg_emit_op(INDEX_op_call);
……
}
```

NICE指令添加

芯来NICE(Nuclei Instruction Co-Unit Extension)

nuclei-sdk/application/baremetal/demo_nice/insn.h

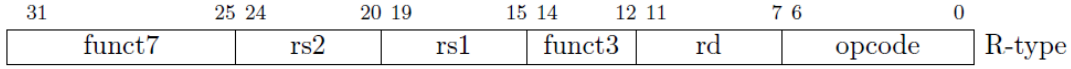



```
/*
 ********************************************************************************************
 * NICE Extension Instruction Format:
 *  .insn and r indicates this is a pseudo and R-type instruction.
 *  0x7b is the value of the opcode field, which means it is a
 *  NICE instruction belonging to custom3.
 * Supported format: only R type here
 * This NICE Demo implements the following 3 instructions for NICE-Core:
 * * CLW or lbuf: Load 12-byte data from memory to row buffer.
 * * CSW or sbuf: Store 12-byte data from row buffer to memory.
 * * CACC or rowsum: Sums a row of the matrix, and columns are accumulated automatically.
 * Supported instructions for this nice demo:
 *  1. custom3 lbuf: burst 4 load(4 cycles) data in memory to row_buf
 *     lbuf (a1)
 *     .insn r opcode, func3, func7, rd, rs1, rs2
 *  2. custom3 sbuf: burst 4 store(4 cycles) row_buf to memory
 *     sbuf (a1)
 *     .insn r opcode, func3, func7, rd, rs1, rs2
 *  3. custom3 acc rowsum: load data from memory(@a1), accumulate row data and write back
 *     rowsum rd, a1, x0(N cycles)
 *     .insn r opcode, func3, func7, rd, rs1, rs2
 ********************************************************************************************
 */
```

```
unsigned int array[ROW_LEN][COL_LEN] = {
        {10, 30, 90},
        {20, 40, 80},
        {30, 90, 120}
    };
```

NICE指令添加

芯来NICE(Nuclei Instruction Co-Unit Extension)




```
lbuf (a1)
 *      .insn r opcode, func3, func7, rd, rs1, rs2
sbuf (a1)
 *      .insn r opcode, func3, func7, rd, rs1, rs2
rowsum rd, a1, x0(N cycles)
 *      .insn r opcode, func3, func7, rd, rs1, rs2
```

```
lbuf (a1)
 *      0000001 00000 ..... 010 00000 1111011
sbuf (a1)
 *      0000010 00000 ..... 010 00000 1111011
rowsum rd, a1, x0(N cycles)
 *      0000110 00000 ..... 110 ..... 1111011
```

nuclei-sdk/application/baremetal/demo_nice/insn.h

```c
#define ROW_LEN     3
#define COL_LEN     3

/** custom nice instruction lbuf */
__STATIC_FORCEINLINE void custom_lbuf(unsigned long* addr)
{
    int zero = 0;

    asm volatile(".insn r 0x7b, 2, 1, x0, %1, x0" : "=r"(zero) : "r"(addr));
}

/** custom nice instruction sbuf */
__STATIC_FORCEINLINE void custom_sbuf(unsigned long* addr)
{
    int zero = 0;

    asm volatile(".insn r 0x7b, 2, 2, x0, %1, x0" : "=r"(zero) : "r"(addr));
}

/** custom nice instruction rowsum */
__STATIC_FORCEINLINE int custom_rowsum(unsigned long* addr)
{
    int rowsum;

    asm volatile(".insn r 0x7b, 6, 6, %0, %1, x0" : "=r"(rowsum) : "r"(addr));

    return rowsum;
}
```

NICE指令之选项添加(可选)

开关选项：-cpu,nuclei-n600,x-nice=true

target/riscv/cpu.h

```
struct RISCVCPU {
    ……
    /* Configuration Settings */
    struct {
        ……
        bool ext_icsr;
        bool ext_nice;
        ……
    } cfg;
};
```

target/riscv/translate.c

```
typedef struct DisasContext {
    DisasContextBase base;
    ……
    int frm;
    bool ext_ifencei;
    bool ext_nice;
    ……
} DisasContext;
```

target/riscv/cpu.c

```
static Property riscv_cpu_properties[] = {
    ……
    DEFINE_PROP_BOOL("u", RISCVCPU, cfg.ext_u, true),
    /* This is experimental so mark with 'x-' */
    DEFINE_PROP_BOOL("x-h", RISCVCPU, cfg.ext_h, false),
    DEFINE_PROP_BOOL("x-v", RISCVCPU, cfg.ext_v, false),
    DEFINE_PROP_BOOL("x-nice", RISCVCPU, cfg.ext_nice, false),
    ……
    DEFINE_PROP_END_OF_LIST(),
};
```

target/riscv/cpu.c

```
static void riscv_tr_init_disas_context(DisasContextBase *dcbase, CPUState *c
s)
{
    ……
    ctx->misa = env->misa;
    ctx->frm = -1;  /* unknown rounding mode */
    ctx->ext_ifencei = cpu->cfg.ext_ifencei;
    ctx->ext_nice = cpu->cfg.ext_nice;
    ……
}
```

misa

NICE指令之DecodeTree

target/riscv/insn32-decode

```
# Fields:
%rs2       20:5
%rs1       15:5
%rd        7:5

# Argument sets:
&r    rd rs1 rs2

# Formats 32:
@r_rs1      ....... ..... .....   ... ..... ....... &r          rs2=0 %rs1 rd=0
@r_rd_rs1   ....... ..... .....   ... ..... ....... &r          rs2=0 %rs1 %rd

# Pattern
custom_lbuf     0000010 00000 ..... 010 00000 1111011  @r_rs1
custom_sbuf   0000100 00000 ..... 010 00000 1111011  @r_rs1
custom_rowsum   0000110 00000 ..... 110 ..... 1111011  @r_rd_rs1
```

```c
static bool trans_custom_1buf(DisasContext *ctx, arg_custom_sbuf *a)
{
    gen_helper_custom_1buf(cpu_env, cpu_gpr[a->rs1]);
    return true;
}

static bool trans_custom_sbuf(DisasContext *ctx, arg_custom_wsetup *a)
{
    gen_helper_custom_sbuf(cpu_env, cpu_gpr[a->rs1]);
    return true;
}

static bool trans_custom_rowsum(DisasContext *ctx, arg_custom_rowsum *a)
{
    // 示例：REQUIRE_NICE;
    gen_helper_custom_rowsum(cpu_gpr[a->rd], cpu_env, cpu_gpr[a->rs1]);
    return true;
}
```

文件:target/riscv/insn_trans/trans_nice.c.inc

```c
#define REQUIRE_NICE do {\
    if (ctx->ext_nice == 0) \
        return false;
        \
} while (0)
```

NICE指令添加之Helper实现

target/riscv/helper.h

```
DEF_HELPER_2(custom_lbuf, void, env, tl)
DEF_HELPER_2(custom_sbuf, void, env, tl)
DEF_HELPER_2(custom_rowsum, tl, env, tl)
```

target/riscv/meson.build

```
riscv_ss.add(files(
  'cpu.c',
  ……
  'nice_helper.c',
  'translate.c',
))
```

target/riscv/nice_helper.c

```
void helper_custom_lbuf(CPURISCVState *env, target_ulong rs1)
{
    for (int i = 0; i < matrix_config; i++) {
        row_buffer[i] = cpu_ldl_le_data(env, rs1 + 4 * i);
    }
}

void helper_custom_sbuf(CPURISCVState *env, target_ulong rs1)
{
    for (int i = 0; i < matrix_config; i++) {
        cpu_stl_le_data(env, rs1 + 4 * i, row_buffer[i]);
    }
}

target_ulong helper_custom_rowsum(CPURISCVState *env, target_ulong rs1)
{
    target_ulong rowsum = 0;
    unsigned int data;
    for (int i = 0; i < matrix_config; i++) {
        data = cpu_ldl_le_data(env, rs1 + 4 * i);
        rowsum += data;
        row_buffer[i] += data;
    }
    return rowsum;
}
```

NICE指令添加之测试(基于老分支)

```
$qemu-system-riscv32  \
-nographic -machine hbird_fpga \
-cpu nuclei-n307fd,x-nice=false \
-kernel ./hbird/demo_nice.elf \
-icount  shift=0 \
-nodefaults -serial stdio
```

支持前：

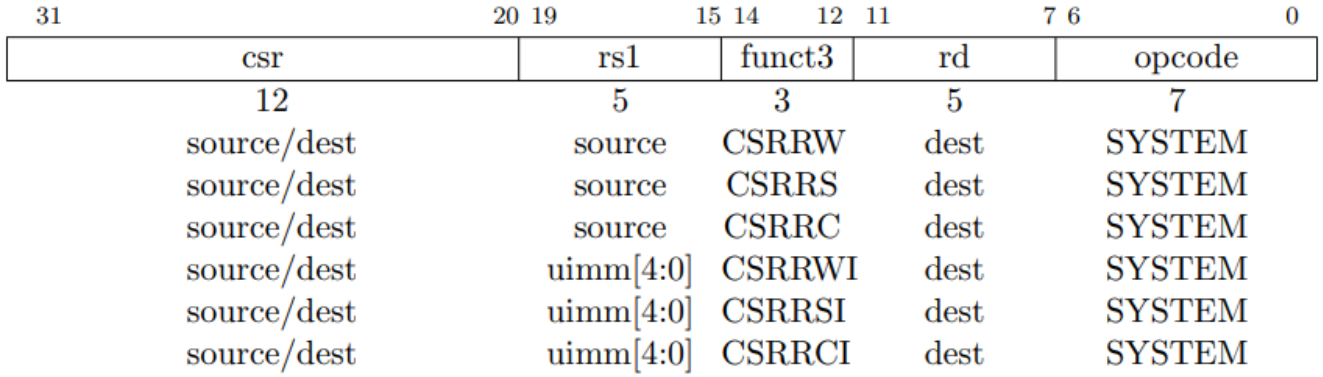
```
Nuclei SDK Build Time: Jun  3 2021, 10:42:40
Download Mode: ILM
CPU Frequency 3277783 Hz

Nuclei Nice Acceleration Demonstration
Warning: This demo required CPU to implement Nuclei provided NICE Demo instructions.
         Otherwise this example will trap to cpu core exception!

1. Print input matrix array
the element of array is :
        10      30      90
        20      40      80
        30      90      120

2. Do reference matrix column sum and row sum
2. Do nice matrix column sum and row sum
MCAUSE : 0x2
MDCAUSE: 0x0
MEPC   : 0x80000b12
MTVAL  : 0x0
```

支持后：

```
Nuclei SDK Build Time: Jun  3 2021, 10:42:40
Download Mode: ILM
CPU Frequency 3277783 Hz

Nuclei Nice Acceleration Demonstration
Warning: This demo required CPU to implement Nuclei provided NICE Demo instructions.
         Otherwise this example will trap to cpu core exception!

1. Print input matrix array
the element of array is :
        10      30      90
        20      40      80
        30      90      120

2. Do reference matrix column sum and row sum
2. Do nice matrix column sum and row sum
3. Compare reference and nice result
  1) Reference result:
the sum of each row is :
                130     140     240
the sum of each col is :
                60      160     290
  2) Nice result:
the sum of each row is :
                130     140     240
the sum of each col is :
                60      160     290
  3) Compare reference vs nice: PASS
4. Performance summary
        normal:
             instret: 556, cycle: 556
        nice  :
             instret: 194, cycle: 185
```

RISC-V CSR寄存器

Control and Status Registers (CSRs)



31 20	19 15	14 12	11 7	6 0
csr	rs1	funct3	rd	opcode
12	5	3	5	7
source/dest	source	CSRRW	dest	SYSTEM
source/dest	source	CSRRS	dest	SYSTEM
source/dest	source	CSRRC	dest	SYSTEM
source/dest	uimm[4:0]	CSRRWI	dest	SYSTEM
source/dest	uimm[4:0]	CSRRSI	dest	SYSTEM
source/dest	uimm[4:0]	CSRRCI	dest	SYSTEM

$riscv-nuclei-elf-objdump -d demo_eclic.elf

```
800002a0:       7ee5d073                csrwi   0x7ee,11
800002a4:       7ef65073                csrwi   0x7ef,12
800002a8:       7eb6d073                csrwi   0x7eb,13
800002ac:       7ed090f3                csrrw   ra,0x7ed,ra
800002b0:       30047073                csrci   mstatus,8
800002b4:       52d2                    lw      t0,52(sp)
800002b6:       7c429073                csrw    0x7c4,t0
800002ba:       52c2                    lw      t0,48(sp)
800002bc:       34129073                csrw    mepc,t0
800002c0:       52b2                    lw      t0,44(sp)
800002c2:       34229073                csrw    mcause,t0
```

CSR Address [11:10]	[9:8]	[7:4]	Hex	Use and Accessibility
User CSRs				
00	00	XXXX	0x000–0x0FF	Standard read/write
01	00	XXXX	0x400–0x4FF	Standard read/write
10	00	XXXX	0x800–0x8FF	Custom read/write
11	00	0XXX	0xC00–0xC7F	Standard read-only
11	00	10XX	0xC80–0xCBF	Standard read-only
11	00	11XX	0xCC0–0xCFF	Custom read-only
Supervisor CSRs				
00	01	XXXX	0x100–0x1FF	Standard read/write
01	01	0XXX	0x500–0x57F	Standard read/write
01	01	10XX	0x580–0x5BF	Standard read/write
01	01	11XX	0x5C0–0x5FF	Custom read/write
10	01	0XXX	0x900–0x97F	Standard read/write
10	01	10XX	0x980–0x9BF	Standard read/write
10	01	11XX	0x9C0–0x9FF	Custom read/write
11	01	0XXX	0xD00–0xD7F	Standard read-only
11	01	10XX	0xD80–0xDBF	Standard read-only
11	01	11XX	0xDC0–0xDFF	Custom read-only
Hypervisor CSRs				
00	10	XXXX	0x200–0x2FF	Standard read/write
01	10	0XXX	0x600–0x67F	Standard read/write
01	10	10XX	0x680–0x6BF	Standard read/write
01	10	11XX	0x6C0–0x6FF	Custom read/write
10	10	0XXX	0xA00–0xA7F	Standard read/write
10	10	10XX	0xA80–0xABF	Standard read/write
10	10	11XX	0xAC0–0xAFF	Custom read/write
11	10	0XXX	0xE00–0xE7F	Standard read-only
11	10	10XX	0xE80–0xEBF	Standard read-only
11	10	11XX	0xEC0–0xEFF	Custom read-only
Machine CSRs				
00	11	XXXX	0x300–0x3FF	Standard read/write
01	11	0XXX	0x700–0x77F	Standard read/write
01	11	100X	0x780–0x79F	Standard read/write
01	11	1010	0x7A0–0x7AF	Standard read/write debug CSRs
01	11	1011	0x7B0–0x7BF	Debug-mode-only CSRs
01	11	11XX	0x7C0–0x7FF	Custom read/write
10	11	0XXX	0xB00–0xB7F	Standard read/write
10	11	10XX	0xB80–0xBBF	Standard read/write
10	11	11XX	0xBC0–0xBFF	Custom read/write
11	11	0XXX	0xF00–0xF7F	Standard read-only
11	11	10XX	0xF80–0xFBF	Standard read-only
11	11	11XX	0xFC0–0xFFF	Custom read-only

来自：Volume 2, Privileged Spec v. 20190608

RISC-V CSR寄存器

target/riscv/insn32.decode

```
# Fields:
%rs1       15:5
%rd        7:5
%csr    20:12
# Formats 32:
@csr     ...........  .....  ... ..... .......          %csr     %rs1 %rd

# *** RV32I Base Instruction Set ***
csrrw    ...........  ..... 001 ..... 1110011 @csr
csrrs    ...........  ..... 010 ..... 1110011 @csr
csrrc    ...........  ..... 011 ..... 1110011 @csr
csrrwi   ...........  ..... 101 ..... 1110011 @csr
csrrsi   ...........  ..... 110 ..... 1110011 @csr
csrrci   ...........  ..... 111 ..... 1110011 @csr
```

定义target/riscv/insn_trans/trans_rvi.c.inc

```
static bool trans_csrrw(DisasContext *ctx, arg_csrrw *a)
static bool trans_csrrs(DisasContext *ctx, arg_csrrs *a)
static bool trans_csrrc(DisasContext *ctx, arg_csrrc *a)
static bool trans_csrrwi(DisasContext *ctx, arg_csrrwi *a)
static bool trans_csrrsi(DisasContext *ctx, arg_csrrsi *a)
static bool trans_csrrci(DisasContext *ctx, arg_csrrci *a)
```

以trans_csrrw为例

```
static bool trans_csrrw(DisasContext *ctx, arg_csrrw *a)
{
    TCGv source1, csr_store, dest, rs1_pass;
    RISCV_OP_CSR_PRE;
    gen_helper_csrrw(dest, cpu_env, source1, csr_store);
    RISCV_OP_CSR_POST;
    return true;
}
```

```
gen_helper_csrrw
gen_helper_csrrs
gen_helper_csrrc
```

```
DEF_HELPER_3(csrrw, tl, env, tl, tl)
DEF_HELPER_4(csrrs, tl, env, tl, tl, tl)
DEF_HELPER_4(csrrc, tl, env, tl, tl, tl)
```

```
helper_csrrw
helper_csrrs
helper_csrrc
```

```
/*
 * riscv_csrrw - read and/or update control and status register
 *
 * csrr   <->  riscv_csrrw(env, csrno, ret_value, 0, 0);
 * csrrw  <->  riscv_csrrw(env, csrno, ret_value, value, -1);
 * csrrs  <->  riscv_csrrw(env, csrno, ret_value, -1, value);
 * csrrc  <->  riscv_csrrw(env, csrno, ret_value, 0, value);
 */

int riscv_csrrw(CPURISCVState *env, int csrno, target_ulong *ret_value,
                target_ulong new_value, target_ulong write_mask)
```

RISC-V CSR寄存器

```
/*
 * riscv_csrrw - read and/or update control and status register
 *
 * csrr   <->  riscv_csrrw(env, csrno, ret_value, 0, 0);
 * csrrw  <->  riscv_csrrw(env, csrno, ret_value, value, -1);
 * csrrs  <->  riscv_csrrw(env, csrno, ret_value, -1, value);
 * csrrc  <->  riscv_csrrw(env, csrno, ret_value, 0, value);
 */

int riscv_csrrw(CPURISCVState *env, int csrno, target_ulong *ret_value,
                target_ulong new_value, target_ulong write_mask)
```

```
/* Control and Status Register function table */
riscv_csr_operations csr_ops[CSR_TABLE_SIZE] = {
    /* User Floating-Point CSRs */
    [CSR_FFLAGS]    = { "fflags",    fs,     read_fflags,  write_fflags },
    [CSR_FRM]       = { "frm",       fs,     read_frm,     write_frm    },
    [CSR_FCSR]      = { "fcsr",      fs,     read_fcsr,    write_fcsr   },
……
}
```

CSR接口：

```
typedef struct {
    const char *name;
    riscv_csr_predicate_fn predicate;
    riscv_csr_read_fn read;
    riscv_csr_write_fn write;
    riscv_csr_op_fn op;
} riscv_csr_operations;
```

RISC-V CSR寄存器

N级别处理器内核自定义**CSR 部分:**

0x7d9	MRW	msaveepc2	自定义寄存器用于保存第二级嵌套 NMI 或异常的 mepc 注意：此寄存器只有配置了两级异常嵌套恢复才会存在
0x7da	MRW	msavecause2	自定义寄存器用于保存第二级嵌套 NMI 或异常的 mcause 注意：此寄存器只有配置了两级异常嵌套恢复才会存在
0x7db	MRW	msavedcause1	自定义寄存器用于保存第一级嵌套异常的 mdcause 注意：此寄存器只有配置了两级异常嵌套恢复才会存在
0x7dc	MRW	msavedcause2	自定义寄存器用于保存第二级嵌套异常的 mdcause 注意：此寄存器只有配置了两级异常嵌套恢复才会存在
0x7eb	MRW	pushmsubm	自定义寄存器用于将 msubm 的值存入堆栈地址空间
0x7ec	MRW	mtvt2	自定义寄存器用于设定非向量中断处理模式的中断入口地址
0x7ed	MRW	jalmnxti	自定义寄存器用于使能 ECLIC 中断，该寄存器的读操作能处理下一个中断同时返回下一个中断 Handler 的入口地址，并跳转至此地址。

来自:《Nuclei_N级别指令架构手册》

CSR接口：

```
typedef struct {
    const char *name;
    riscv_csr_predicate_fn predicate;
    riscv_csr_read_fn read;
    riscv_csr_write_fn write;
    riscv_csr_op_fn op;
} riscv_csr_operations;
```

RISC-V CSR寄存器

target/riscv/cpu_bits.h

```
#define CSR_PUSHMSUBM      0x07eb
#define CSR_MTVT2      0x07ec
#define CSR_JALMNXTI      0x07ed
#define CSR_PUSHMCAUSE      0x07ee
#define CSR_PUSHMEPC      0x07ef
```

target/riscv/csr.c

```
/* Control and Status Register function table */
riscv_csr_operations csr_ops[CSR_TABLE_SIZE] = {
    /* User Floating-Point CSRs */
    [CSR_FFLAGS]    = { "fflags",    fs,     read_fflags,  write_fflags },
    [CSR_FRM]       = { "frm",       fs,     read_frm,     write_frm    },
    [CSR_FCSR]      = { "fcsr",      fs,     read_fcsr,    write_fcsr   },
    /* Vector CSRs */
    [CSR_VSTART]    = { "vstart",    vs,     read_vstart,  write_vstart },
    [CSR_VXSAT]     = { "vxsat",     vs,     read_vxsat,   write_vxsat  },
    [CSR_PUSHMCAUSE]     = { “pushmcause",     eclic,     NULL,   NULL, rmw_pushmcause },
    ……
}
```

```
struct  riscv_csr_operations
const char *name;
riscv_csr_predicate_fn predicate
riscv_csr_read_fn read
riscv_csr_write_fn write
riscv_csr_op_fn op
```

CSR PUSHMCAUSE:

处理器 定义了通过 pushmcause 寄存器 csrrwi 操作实现的 CSR 指令，存储 mcause 的值到堆栈指针作为基地址的memory 空间

以如下指令为例介绍此CSR指令：

csrrwi x0，PUSHMCAUSE，1

该指令的操作是将mcause寄存器的值存到SP（堆栈指针）+1*4的地址。

```
static int rmw_pushmcause(CPURISCVState *env, int csrno, target_ulong *ret_value,
                target_ulong new_value, target_ulong write_mask)
{
    uint64_t notify_addr = new_value * 4 + env->gpr[2];
    ……
    cpu_physical_memory_rw(notify_addr, &env->mcause,  4, 1);
    ……
    return 0;
}
```

下节课内容：

中断虚拟化

- RISCV IRQ实现分析
- Nuclei Timer介绍与实现
- Nuclei Eclic介绍与实现
- clic clint plic相关实现对比

谢谢

wangjunqiang@iscas.ac.cn