从零开始的RISC-V模拟器开发

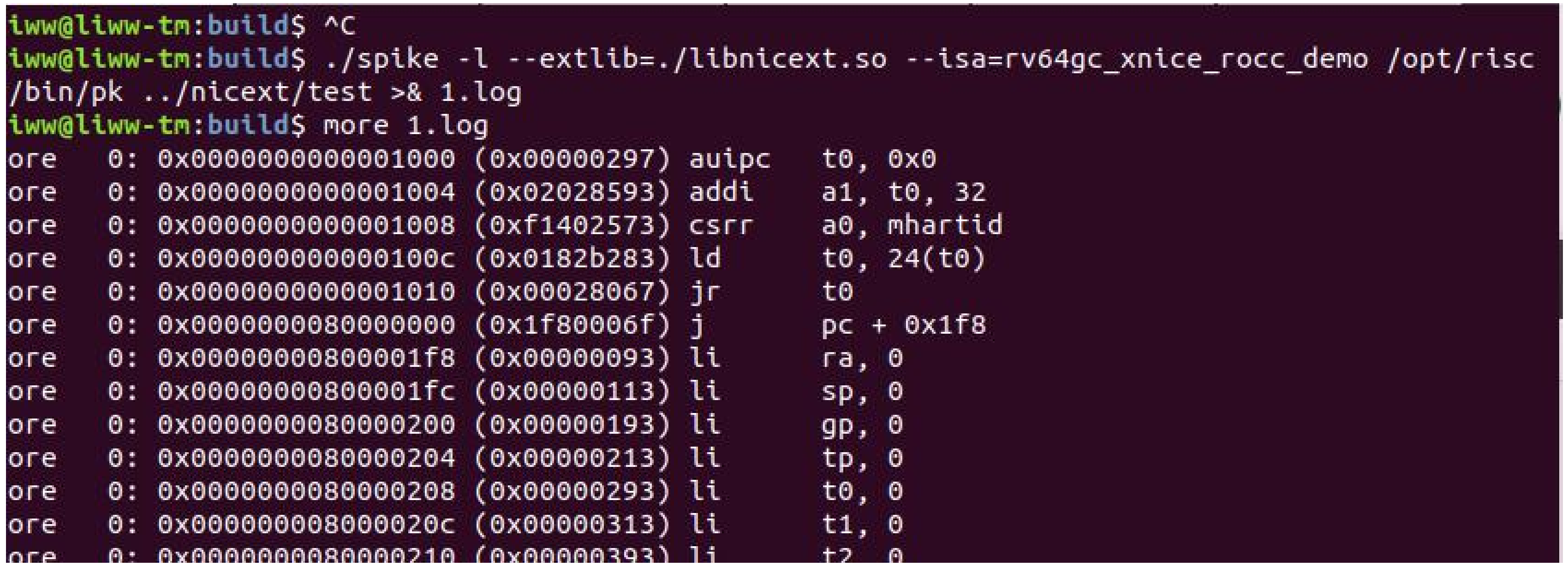
第3讲 Spike篇之Debug机制

中国科学院软件研究所
PLCT实验室

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

回顾

类型	命令	描述
Log选项	-l	Generate a log of execution
交互调试	-d	Interactive debug mode



```
iww@liww-tm:build$ ^C
iww@liww-tm:build$ ./spike -l --extlib=./libnicext.so --isa=rv64gc_xnice_rocc_demo /opt/risc
/bin/pk ../nicext/test >& 1.log
iww@liww-tm:build$ more 1.log
ore   0: 0x0000000000001000 (0x00000297) auipc   t0, 0x0
ore   0: 0x0000000000001004 (0x02028593) addi    a1, t0, 32
ore   0: 0x0000000000001008 (0xf1402573) csrr    a0, mhartid
ore   0: 0x000000000000100c (0x0182b283) ld      t0, 24(t0)
ore   0: 0x0000000000001010 (0x00028067) jr      t0
ore   0: 0x0000000080000000 (0x1f80006f) j       pc + 0x1f8
ore   0: 0x00000000800001f8 (0x00000093) li      ra, 0
ore   0: 0x00000000800001fc (0x00000113) li      sp, 0
ore   0: 0x0000000080000200 (0x00000193) li      gp, 0
ore   0: 0x0000000080000204 (0x00000213) li      tp, 0
ore   0: 0x0000000080000208 (0x00000293) li      t0, 0
ore   0: 0x000000008000020c (0x00000313) li      t1, 0
ore   0: 0x0000000080000210 (0x00000393) li      t2, 0
```

回顾

```
class processor_t : public abstract_device_t
{
  ...
  std::unordered_map<std::string, extension_t*>
custom_extensions;
  disassembler_t* disassembler;
  state_t state;
  ...
  std::vector<bool> extension_table;
  ...
  std::vector<insn_desc_t> instructions;


  static const size_t OPCODE_CACHE_SIZE = 8191;
  insn_desc_t opcode_cache[OPCODE_CACHE_SIZE];
  ...
  vectorUnit_t VU;
}
```

```
void processor_t::register_extension(extension_t* x)
{
  for (auto insn : x->get_instructions())
    register_insn(insn);
  build_opcode_map();

  if (disassembler)
    for (auto disasm_insn : x->get_disasms())
      disassembler->add_insn(disasm_insn);

  if (!custom_extensions.insert(std::make_pair(x->name(), x)).second) {
    fprintf(stderr, "extensions must have unique names (got two named
\"%s\"!)\n", x->name());
    abort();
  }

  x->set_processor(this);
}
```

中国科学院软件研究所 Institute of Software Chinese Academy of Sciences | 智能软件研究中心 Intelligent Software Research Center

汇编器结构

riscv/disasm.h

```
class disassembler_t
{
 public:
  disassembler_t(int xlen);
  ~disassembler_t();

  std::string disassemble(insn_t insn) const;
  const disasm_insn_t* lookup(insn_t insn) const;

  void add_insn(disasm_insn_t* insn);

 private:
  static const int HASH_SIZE = 256;
  std::vector<const disasm_insn_t*> chain[HASH_SIZE+1];
};
```

汇编器结构

riscv/disasm.h

```
class disassembler_t
{
 public:
  disassembler_t(int xlen);
  ~disassembler_t();

  std::string disassemble(insn_t insn) const;
  const disasm_insn_t* lookup(insn_t insn) const;

  void add_insn(disasm_insn_t* insn);

 private:
  static const int HASH_SIZE = 256;
  std::vector<const disasm_insn_t*> chain[HASH_SIZE+1];
};
```

disasm/disasm.cc

```
void NOINLINE disassembler_t::add_insn(disasm_insn_t* insn)
{
  size_t idx = HASH_SIZE;
  if (insn->get_mask() % HASH_SIZE == HASH_SIZE - 1)
    idx = insn->get_match() % HASH_SIZE;
  chain[idx].push_back(insn);
}
```

指令集汇编注册

riscv/
processor.cc

```
processor_t::processor_t(const char* isa, const char* priv, const char* varch,
                         simif_t* sim, uint32_t id, bool halt_on_reset,
                         FILE* log_file)
  : debug(false), halt_request(HR_NONE), sim(sim), id(id), xlen(0),
  histogram_enabled(false), log_commits_enabled(false),
  log_file(log_file), halt_on_reset(halt_on_reset),
  extension_table(256, false), impl_table(256, false), last_pc(1), executions(1)
{
  VU.p = this;

  parse_isa_string(isa);
  parse_priv_string(priv);
  parse_varch_string(varch);

  register_base_instructions();
  mmu = new mmu_t(sim, this);

  disassembler = new disassembler_t(max_xlen);
  for (auto e : custom_extensions)
    for (auto disasm_insn : e.second->get_disasms())
      disassembler->add_insn(disasm_insn);
  ...
}
```

指令集汇编注册—基础指令集注册

```
disassembler_t::disassembler_t(int xlen)
{
 ...
  #define DECLARE_INSN(code, match, mask) \
   const uint32_t match_##code = match; \
   const uint32_t mask_##code = mask;
  #define DECLARE_RV32_ONLY(code) {}
  #define DECLARE_RV64_ONLY(code) {}
  #include "encoding.h"
  #undef DECLARE_RV64_INSN
  #undef DECLARE_RV32_INSN
  #undef DECLARE_INSN

  // explicit per-instruction disassembly
  #define DISASM_INSN(name, code, extra, ...) \
    add_insn(new disasm_insn_t(name, match_##code, mask_##code |
(extra), __VA_ARGS__));
  #define DEFINE_NOARG(code) \
    add_insn(new disasm_insn_t(#code, match_##code, mask_##code,
{}));
  ...
  DISASM_INSN("c.ebreak", c_add, mask_rd | mask_rvc_rs2, {});
```

```
add_insn(new disasm_insn_t("ret", match_c_jr | match_rd_ra, mask_c_jr |
mask_rd | mask_rvc_imm, {}));
  DISASM_INSN("c.jr", c_jr, mask_rvc_imm, {&rvc_rs1});
  DISASM_INSN("c.jalr", c_jalr, mask_rvc_imm, {&rvc_rs1});
  ...
  // provide a default disassembly for all instructions as a fallback
  #define DECLARE_INSN(code, match, mask) \
   add_insn(new disasm_insn_t(#code " (args unknown)", match, mask, {}));
  #define DECLARE_RV32_ONLY(code) {}
  #define DECLARE_RV64_ONLY(code) {}
  #include "encoding.h"
  #undef DECLARE_RV64_INSN
  #undef DECLARE_RV32_INSN
  #undef DECLARE_INSN
}
```

指令汇编结构

```
class disasm_insn_t
{
 public:
  NOINLINE disasm_insn_t(const char* name, uint32_t match, uint32_t mask,
                         const std::vector<const arg_t*>& args)
    : match(match), mask(mask), args(args), name(strdup(name)) {}
  ~disasm_insn_t() { free(const_cast<char *>(name)); }

  bool operator == (insn_t insn) const

  const char* get_name() const

  std::string to_string(insn_t insn) const

  uint32_t get_match() const { return match; }
  uint32_t get_mask() const { return mask; }

 private:
  uint32_t match;
  uint32_t mask;
  std::vector<const arg_t*> args;
  const char* name;
};
```

```
class arg_t
{
 public:
  virtual std::string to_string(insn_t val) const = 0;
  virtual ~arg_t() {}
};
```

指令汇编表示

```
std::string to_string(insn_t insn) const
  {
    std::stringstream s;
    int len;
    for (len = 0; name[len]; len++)
      s << (name[len] == '_' ? '.' : name[len]);

    if (args.size())
    {
      bool next_arg_optional  = false;
      s << std::string(std::max(1, 8 - len), ' ');
      for (size_t i = 0; i < args.size(); i++) {
        if (args[i] == &opt) {
          next_arg_optional = true;
          continue;
        }
```

```
        std::string argString = args[i]->to_string(insn);
        if (next_arg_optional) {
          next_arg_optional = false;
          if (argString.empty()) continue;
        }
        if (i != 0) s << ", ";
        s << argString;
      }
    }
    return s.str();
  }
```

insn_name src1，src2，...

常用的args

disasm/disasm.cc

```
struct : public arg_t {
  std::string to_string(insn_t insn) const {
    return std::to_string((int)insn.i_imm()) + '(' +
xpr_name[insn.rs1()] + ')';
  }
} load_address;
```

```
struct : public arg_t {
  std::string to_string(insn_t insn) const {
    return xpr_name[insn.rd()];
  }
} xrd;
```

```
struct : public arg_t {
  std::string to_string(insn_t insn) const {
    return fpr_name[insn.rd()];
  }
} frd;
```

```
struct : public arg_t {
  std::string to_string(insn_t insn) const {
    switch (insn.csr())
    {
      #define DECLARE_CSR(name, num) case num: return #name;
      #include "encoding.h"
      #undef DECLARE_CSR
      default:
      {
        char buf[16];
        snprintf(buf, sizeof buf, "unknown_%03" PRIx64, insn.csr());
        return std::string(buf);
      }
    }
  }
} csr;
```

```
struct : public arg_t {
  std::string to_string(insn_t insn) const {
    return std::to_string((int)insn.i_imm());
  }
} imm;
```

寄存器名称

```
const char* xpr_name[] = {
  "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"
};

const char* fpr_name[] = {
  "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"
};

const char* vr_name[] = {
  "v0",  "v1",  "v2",  "v3",  "v4",  "v5",  "v6",  "v7",
  "v8",  "v9",  "v10", "v11", "v12", "v13", "v14", "v15",
  "v16", "v17", "v18", "v19", "v20", "v21", "v22", "v23",
  "v24", "v25", "v26", "v27", "v28", "v29", "v30", "v31"
};
```

disasm/regname.cc

```
const char* csr_name(int which) {
  switch (which) {
    #define DECLARE_CSR(name, number)  case number:
return #name;
    #include "encoding.h"
    #undef DECLARE_CSR
  }
  return "unknown-csr";
}
```

```
DECLARE_CSR(fflags, CSR_FFLAGS)
DECLARE_CSR(frm, CSR_FRM)
DECLARE_CSR(fcsr, CSR_FCSR)
DECLARE_CSR(ustatus, CSR_USTATUS)
DECLARE_CSR(uie, CSR_UIE)
DECLARE_CSR(utvec, CSR_UTVEC)
DECLARE_CSR(vstart, CSR_VSTART)
```

基础扩展指令添加汇编支持

disasm/disasm.cc

```
disassembler_t::disassembler_t(int xlen)
{
  ...
    add_insn(new disasm_insn_t("custom_sbuf", 0x0400207b, 0xFE00707F, {&xrs1}));
    add_insn(new disasm_insn_t("custom_wsetup", 0x0800207b, 0xFE00707F, {&xrs1}));
    add_insn(new disasm_insn_t("custom_rowsum", 0x0C00607b, 0xFE00707F, {&xrd, &xrs1}));

  // provide a default disassembly for all instructions as a fallback
  #define DECLARE_INSN(code, match, mask) \
   add_insn(new disasm_insn_t(#code " (args unknown)", match, mask, {}));
  #define DECLARE_RV32_ONLY(code) {}
  #define DECLARE_RV64_ONLY(code) {}
  #include "encoding.h"
  #undef DECLARE_RV64_INSN
  #undef DECLARE_RV32_INSN
  #undef DECLARE_INSN
}
```

外部扩展指令添加汇编支持

nicext/nice_demo.cc

```
class nice_demo_t : public extension_t
{
 public:
  const char* name() { return "nice_demo"; }

  nice_demo_t() {}

  std::vector<insn_desc_t> get_instructions() {
    std::vector<insn_desc_t> insns;
    insns.push_back((insn_desc_t){0x0400207b, 0xFE00707F, custom_sbuf, custom_sbuf});
    insns.push_back((insn_desc_t){0x0800207b, 0xFE00707F, custom_wsetup, custom_wsetup});
    insns.push_back((insn_desc_t){0x0C00607b, 0xFE00707F, custom_rowsum, custom_rowsum});
    return insns;
  }

  std::vector<disasm_insn_t*> get_disasms() {
    std::vector<disasm_insn_t*> insns;
    insns.push_back(new disasm_insn_t("custom_sbuf", 0x0400207b, 0xFE00707F, {&xrs1}));
    insns.push_back(new disasm_insn_t("custom_wsetup", 0x0800207b, 0xFE00707F, {&xrs1}));
    insns.push_back(new disasm_insn_t("custom_rowsum", 0x0C00607b, 0xFE00707F, {&xrd,
&xrs1}));
    return insns;
  }
};
```

```
struct : public arg_t {
  std::string to_string(insn_t insn) const {
    return xpr_name[insn.rs1()];
  }
} xrs1;

struct : public arg_t {
  std::string to_string(insn_t insn) const {
    return xpr_name[insn.rd()];
  }
} xrd;
```

外部扩展指令Rocc汇编支持

riscv/rocc.h

```
class rocc_t : public extension_t
{
 public:
  virtual reg_t custom0(rocc_insn_t insn, reg_t xs1, reg_t xs2);
  virtual reg_t custom1(rocc_insn_t insn, reg_t xs1, reg_t xs2);
  virtual reg_t custom2(rocc_insn_t insn, reg_t xs1, reg_t xs2);
  virtual reg_t custom3(rocc_insn_t insn, reg_t xs1, reg_t xs2);
  std::vector<insn_desc_t> get_instructions();
  std::vector<disasm_insn_t*> get_disasms();
};
```

riscv/rocc.cc

```
std::vector<insn_desc_t> rocc_t::get_instructions()
{
  std::vector<insn_desc_t> insns;
  insns.push_back((insn_desc_t){0x0b, 0x7f, &::illegal_instruction, c0});
  insns.push_back((insn_desc_t){0x2b, 0x7f, &::illegal_instruction, c1});
  insns.push_back((insn_desc_t){0x5b, 0x7f, &::illegal_instruction, c2});
  insns.push_back((insn_desc_t){0x7b, 0x7f, &::illegal_instruction, c3});
  return insns;
}

std::vector<disasm_insn_t*> rocc_t::get_disasms()
{
  std::vector<disasm_insn_t*> insns;
  return insns;
}
```

Rocc汇编支持

```
disassembler_t::disassembler_t(int xlen)
{
  ...
  // provide a default disassembly for all instructions as a fallback
  #define DECLARE_INSN(code, match, mask) \
   add_insn(new disasm_insn_t(#code " (args unknown)", match, mask,
{}));
  #define DECLARE_RV32_ONLY(code) {}
  #define DECLARE_RV64_ONLY(code) {}
  #include "encoding.h"
  #undef DECLARE_RV64_INSN
  #undef DECLARE_RV32_INSN
  #undef DECLARE_INSN
}
```

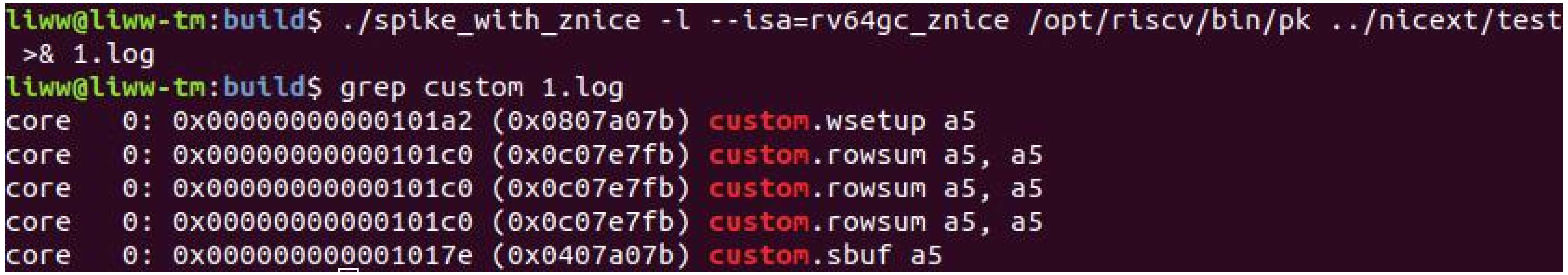
riscv/encoding.h

```
DECLARE_INSN(custom3, MATCH_CUSTOM3, MASK_CUSTOM3)
DECLARE_INSN(custom3_rs1, MATCH_CUSTOM3_RS1,
MASK_CUSTOM3_RS1)
DECLARE_INSN(custom3_rs1_rs2, MATCH_CUSTOM3_RS1_RS2,
MASK_CUSTOM3_RS1_RS2)
DECLARE_INSN(custom3_rd, MATCH_CUSTOM3_RD,
MASK_CUSTOM3_RD)
DECLARE_INSN(custom3_rd_rs1, MATCH_CUSTOM3_RD_RS1,
MASK_CUSTOM3_RD_RS1)
DECLARE_INSN(custom3_rd_rs1_rs2,
MATCH_CUSTOM3_RD_RS1_RS2, MASK_CUSTOM3_RD_RS1_RS2)
```

汇编支持优先级： **disassembler_t:add_insn > encoding.h: DECLARE_INSN > extension: get_disasms**

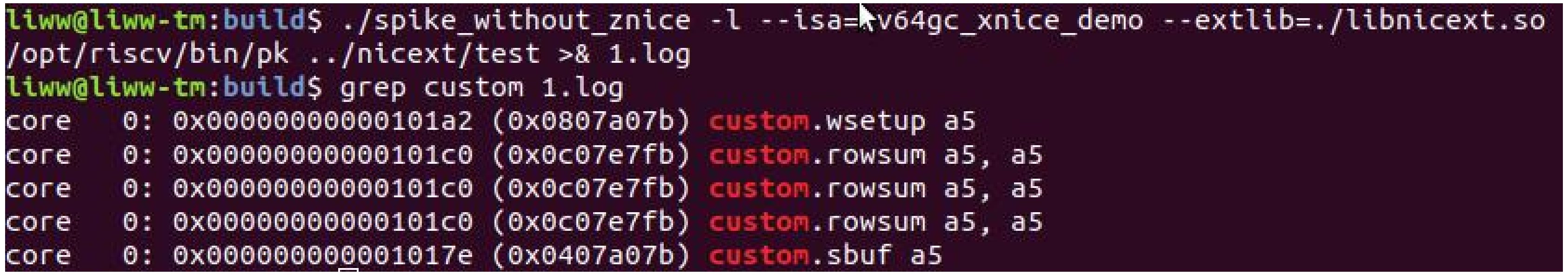
如果希望**nicext_demo**的汇编支持起作用，需要注释掉这些指令声明以及基础指令扩展的汇编支持

基础扩展指令添加汇编支持测试



```
liww@liww-tm:build$ ./spike_with_znice -l --isa=rv64gc_znice /opt/riscv/bin/pk ../nicext/test
 >& 1.log
liww@liww-tm:build$ grep custom 1.log
core   0: 0x00000000000101a2 (0x0807a07b) custom.wsetup a5
core   0: 0x00000000000101c0 (0x0c07e7fb) custom.rowsum a5, a5
core   0: 0x00000000000101c0 (0x0c07e7fb) custom.rowsum a5, a5
core   0: 0x00000000000101c0 (0x0c07e7fb) custom.rowsum a5, a5
core   0: 0x000000000001017e (0x0407a07b) custom.sbuf a5
```

外部扩展指令汇编支持测试

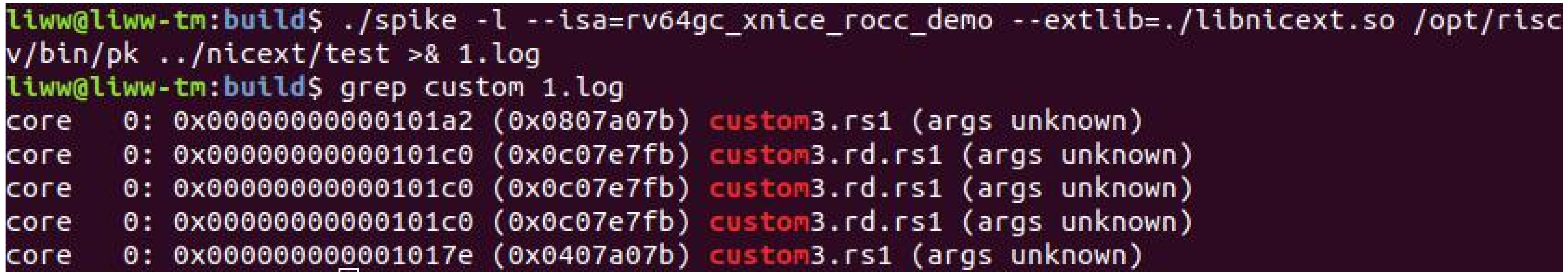


```
liww@liww-tm:build$ ./spike_without_znice -l --isa=rv64gc_xnice_demo --extlib=./libnicext.so
/opt/riscv/bin/pk ../nicext/test >& 1.log
liww@liww-tm:build$ grep custom 1.log
core   0: 0x00000000000101a2 (0x0807a07b) custom.wsetup a5
core   0: 0x00000000000101c0 (0x0c07e7fb) custom.rowsum a5, a5
core   0: 0x00000000000101c0 (0x0c07e7fb) custom.rowsum a5, a5
core   0: 0x00000000000101c0 (0x0c07e7fb) custom.rowsum a5, a5
core   0: 0x000000000001017e (0x0407a07b) custom.sbuf a5
```

```
//DECLARE_INSN(custom3, MATCH_CUSTOM3, MASK_CUSTOM3)
//DECLARE_INSN(custom3_rs1, MATCH_CUSTOM3_RS1, MASK_CUSTOM3_RS1)
//DECLARE_INSN(custom3_rs1_rs2, MATCH_CUSTOM3_RS1_RS2, MASK_CUSTOM3_RS1_RS2)
//DECLARE_INSN(custom3_rd, MATCH_CUSTOM3_RD, MASK_CUSTOM3_RD)
//DECLARE_INSN(custom3_rd_rs1, MATCH_CUSTOM3_RD_RS1, MASK_CUSTOM3_RD_RS1)
//DECLARE_INSN(custom3_rd_rs1_rs2, MATCH_CUSTOM3_RD_RS1_RS2,
MASK_CUSTOM3_RD_RS1_RS2)
//DECLARE_INSN(custom_sbuf, MATCH_CUSTOM_SBUF, MASK_CUSTOM_SBUF)
//DECLARE_INSN(custom_wsetup, MATCH_CUSTOM_WSETUP, MASK_CUSTOM_WSETUP)
//DECLARE_INSN(custom_rowsum, MATCH_CUSTOM_ROWSUM, MASK_CUSTOM_ROWSUM)
```

```
  //add_insn(new disasm_insn_t("custom_sbuf", 0x0400207b, 0xFE00707F, {&xrs1}));
  //add_insn(new disasm_insn_t("custom_wsetup", 0x0800207b, 0xFE00707F, {&xrs1}));
  //add_insn(new disasm_insn_t("custom_rowsum", 0x0C00607b, 0xFE00707F, {&xrd,
&xrs1}));
```

Rocc汇编支持测试



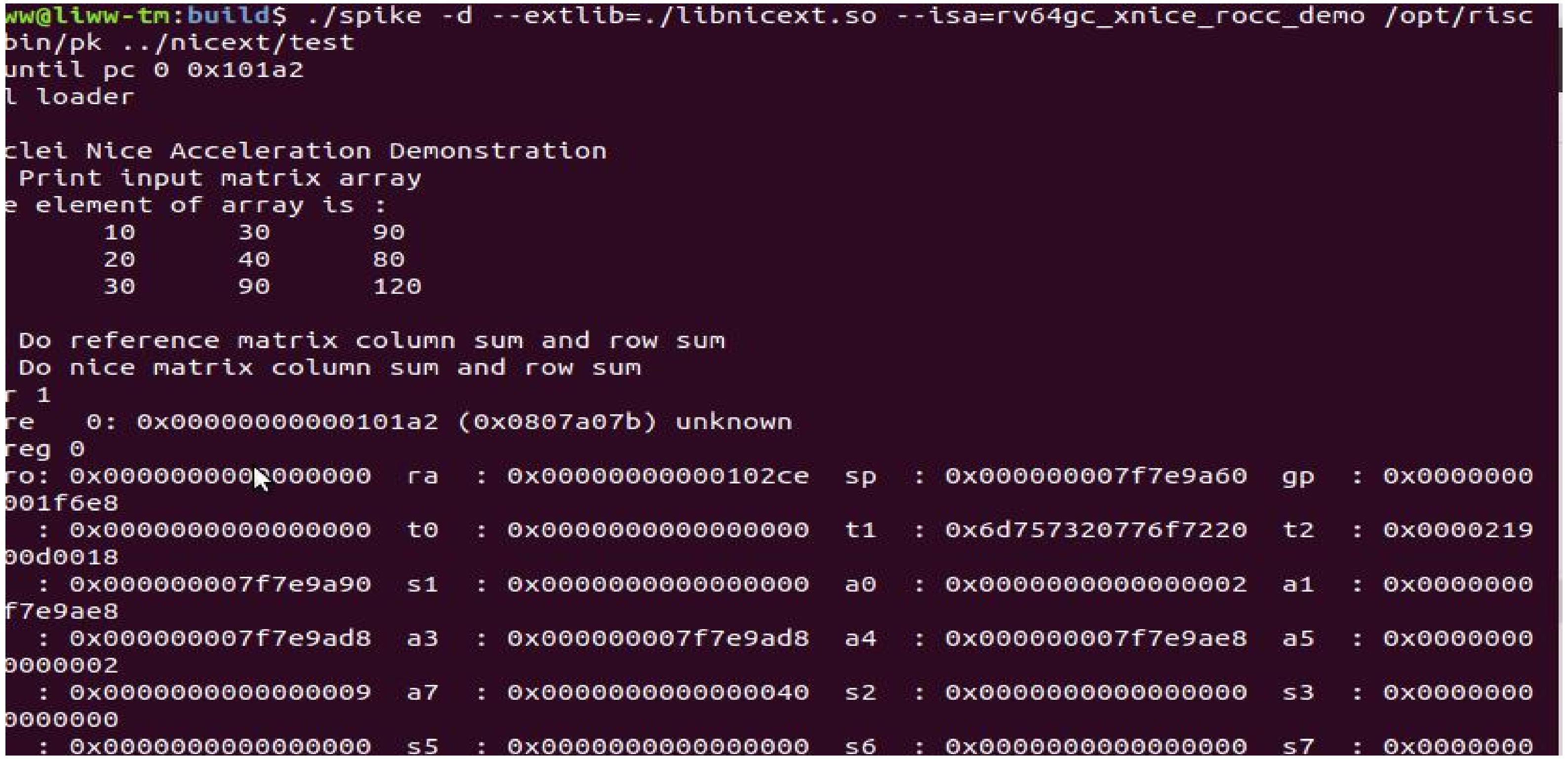
```
liww@liww-tm:build$ ./spike -l --isa=rv64gc_xnice_rocc_demo --extlib=./libnicext.so /opt/risc
v/bin/pk ../nicext/test >& 1.log
liww@liww-tm:build$ grep custom 1.log
core   0: 0x00000000000101a2 (0x0807a07b) custom3.rs1 (args unknown)
core   0: 0x00000000000101c0 (0x0c07e7fb) custom3.rd.rs1 (args unknown)
core   0: 0x00000000000101c0 (0x0c07e7fb) custom3.rd.rs1 (args unknown)
core   0: 0x00000000000101c0 (0x0c07e7fb) custom3.rd.rs1 (args unknown)
core   0: 0x000000000001017e (0x0407a07b) custom3.rs1 (args unknown)
```

```
DECLARE_INSN(custom3, MATCH_CUSTOM3, MASK_CUSTOM3)
DECLARE_INSN(custom3_rs1, MATCH_CUSTOM3_RS1, MASK_CUSTOM3_RS1)
DECLARE_INSN(custom3_rs1_rs2, MATCH_CUSTOM3_RS1_RS2, MASK_CUSTOM3_RS1_RS2)
DECLARE_INSN(custom3_rd, MATCH_CUSTOM3_RD, MASK_CUSTOM3_RD)
DECLARE_INSN(custom3_rd_rs1, MATCH_CUSTOM3_RD_RS1, MASK_CUSTOM3_RD_RS1)
DECLARE_INSN(custom3_rd_rs1_rs2, MATCH_CUSTOM3_RD_RS1_RS2,
MASK_CUSTOM3_RD_RS1_RS2)
//DECLARE_INSN(custom_sbuf, MATCH_CUSTOM_SBUF, MASK_CUSTOM_SBUF)
//DECLARE_INSN(custom_wsetup, MATCH_CUSTOM_WSETUP, MASK_CUSTOM_WSETUP)
//DECLARE_INSN(custom_rowsum, MATCH_CUSTOM_ROWSUM, MASK_CUSTOM_ROWSUM)
```

```
  //add_insn(new disasm_insn_t("custom_sbuf", 0x0400207b, 0xFE00707F, {&xrs1}));
  //add_insn(new disasm_insn_t("custom_wsetup", 0x0800207b, 0xFE00707F, {&xrs1}));
  //add_insn(new disasm_insn_t("custom_rowsum", 0x0C00607b, 0xFE00707F, {&xrd,
&xrs1}));
```

交互调试选项

类型	命令	描述
交互调试	-d	Interactive debug mode



```
ww@liww-tm:build$ ./spike -d --extlib=./libnicext.so --isa=rv64gc_xnice_rocc_demo /opt/risc
bin/pk ../nicext/test
until pc 0 0x101a2
l loader

clei Nice Acceleration Demonstration
 Print input matrix array
e element of array is :
      10       30       90
      20       40       80
      30       90       120

 Do reference matrix column sum and row sum
 Do nice matrix column sum and row sum
r 1
re   0: 0x00000000000101a2 (0x0807a07b) unknown
reg 0
ro: 0x0000000000000000  ra  : 0x00000000000102ce  sp  : 0x000000007f7e9a60  gp  : 0x0000000
001f6e8
  : 0x0000000000000000  t0  : 0x0000000000000000  t1  : 0x6d757320776f7220  t2  : 0x0000219
00d0018
  : 0x000000007f7e9a90  s1  : 0x0000000000000000  a0  : 0x0000000000000002  a1  : 0x0000000
f7e9ae8
  : 0x000000007f7e9ad8  a3  : 0x000000007f7e9ad8  a4  : 0x000000007f7e9ae8  a5  : 0x0000000
0000002
  : 0x0000000000000009  a7  : 0x0000000000000040  s2  : 0x0000000000000000  s3  : 0x0000000
0000000
  : 0x0000000000000000  s5  : 0x0000000000000000  s6  : 0x0000000000000000  s7  : 0x0000000
```

交互调试命令

```
"Interactive commands:\n"
"reg <core> [reg]              # Display [reg] (all if omitted) in <core>\n"
"fregh <core> <reg>            # Display half precision <reg> in <core>\n"
"fregs <core> <reg>            # Display single precision <reg> in <core>\n"
"fregd <core> <reg>            # Display double precision <reg> in <core>\n"
"vreg <core> [reg]             # Display vector [reg] (all if omitted) in <core>\n"
"pc <core>                     # Show current PC in <core>\n"
"mem <hex addr>                # Show contents of physical memory\n"
"str <core> <hex addr>         # Show NUL-terminated C string at <hex addr> in core <core>\n"
"until reg <core> <reg> <val>  # Stop when <reg> in <core> hits <val>\n"
"until pc <core> <val>         # Stop when PC in <core> hits <val>\n"
"untiln pc <core> <val>        # Run noisy and stop when PC in <core> hits <val>\n"
"until mem <addr> <val>        # Stop when memory <addr> becomes <val>\n"
"while reg <core> <reg> <val>  # Run while <reg> in <core> is <val>\n"
"while pc <core> <val>         # Run while PC in <core> is <val>\n"
"while mem <addr> <val>        # Run while memory <addr> is <val>\n"
"run [count]                   # Resume noisy execution (until CTRL+C, or [count] insns)\n"
"r [count]                       Alias for run\n"
"rs [count]                    # Resume silent execution (until CTRL+C, or [count] insns)\n"
"quit                          # End the simulation\n"
"q                               Alias for quit\n"
"help                          # This screen!\n"
"h                               Alias for help\n"
"Note: Hitting enter is the same as: run 1\n"
```

交互调试触发流程

spike_main/spike.cc

```
parser.option('d', 0, 0, [&](const char* s){debug = true;});
...
s.set_debug(debug);
```

```
void sim_t::set_debug(bool value)
{
  debug = value;
}
```

```
void sim_t::main()
{
  if (!debug && log)
    set_procs_debug(true);

  while (!done())
  {
    if (debug || ctrlc_pressed)
      interactive();
    else
      step(INTERLEAVE);
    if (remote_bitbang) {
      remote_bitbang->tick();
    }
  }
}
```

交互调试命令入口

riscv/interactive.cc

```
void sim_t::interactive()
{
  typedef void (sim_t::*interactive_func)(const std::string&, const
std::vector<std::string>&);
  std::map<std::string,interactive_func> funcs;

  funcs["run"] = &sim_t::interactive_run_noisy;
  funcs["r"] = funcs["run"];
  funcs["rs"] = &sim_t::interactive_run_silent;
  funcs["vreg"] = &sim_t::interactive_vreg;
  funcs["reg"] = &sim_t::interactive_reg;
  funcs["freg"] = &sim_t::interactive_freg;
  funcs["fregh"] = &sim_t::interactive_fregh;
  funcs["fregs"] = &sim_t::interactive_fregs;
  funcs["fregd"] = &sim_t::interactive_fregd;
  funcs["pc"] = &sim_t::interactive_pc;
  funcs["mem"] = &sim_t::interactive_mem;
  funcs["str"] = &sim_t::interactive_str;
  funcs["until"] = &sim_t::interactive_until_silent;
  funcs["untiln"] = &sim_t::interactive_until_noisy;
  funcs["while"] = &sim_t::interactive_until_silent;
  funcs["quit"] = &sim_t::interactive_quit;
  funcs["q"] = funcs["quit"];
  funcs["help"] = &sim_t::interactive_help;
  funcs["h"] = funcs["help"];
```

```
while (!done())
 {
  std::cerr << ": " << std::flush;
  std::string s = readline(2);

  std::stringstream ss(s);
  std::string cmd, tmp;
  std::vector<std::string> args;

  if (!(ss >> cmd))
  {
   set_procs_debug(true);
   step(1);
   continue;
  }
  while (ss >> tmp)
   args.push_back(tmp);
  try
  {
   if(funcs.count(cmd))
    (this->*funcs[cmd])(cmd, args);
   else
    fprintf(stderr, "Unknown command %s\n", cmd.c_str());
  }
  catch(trap_t& t) {}
 }
 ctrlc_pressed = false;
}
```

交互调试命令—reg

```
funcs["reg"] = &sim_t::interactive_reg;
```

```
void sim_t::interactive_reg(const std::string& cmd, const
std::vector<std::string>& args)
{
 if (args.size() == 1) {
   // Show all the regs!
   processor_t *p = get_core(args[0]);

   for (int r = 0; r < NXPR; ++r) {
     fprintf(stderr, "%-4s: 0x%016" PRIx64 "  ",
xpr_name[r], p->get_state()->XPR[r]);
     if ((r + 1) % 4 == 0)
       fprintf(stderr, "\n");
   }
 } else
   fprintf(stderr, "0x%016" PRIx64 "\n", get_reg(args));
}
```

```
reg_t sim_t::get_reg(const std::vector<std::string>& args)
{
  if(args.size() != 2)
    throw trap_interactive();

  processor_t *p = get_core(args[0]);
  unsigned long r = std::find(xpr_name, xpr_name + NXPR, args[1]) - xpr_name;
  if (r == NXPR) {
    char *ptr;
    r = strtoul(args[1].c_str(), &ptr, 10);
    if (*ptr) {
      #define DECLARE_CSR(name, number) if (args[1] == #name) return p-
>get_csr(number);
      #include "encoding.h"          // generates if's for all csrs
      r = NXPR;                 // else case (csr name not found)
      #undef DECLARE_CSR
    }
  }
  if (r >= NXPR)
    throw trap_interactive();

  return p->get_state()->XPR[r];
}
```

交互调试命令—until

```
funcs["until"] = &sim_t::interactive_until_silent;
```

```
void sim_t::interactive_until_silent(const std::string& cmd, const
std::vector<std::string>& args)
{
  interactive_until(cmd, args, false);
}
```

```
void sim_t::interactive_until(const std::string& cmd, const
std::vector<std::string>& args, bool noisy)
{
  bool cmd_until = cmd == "until" || cmd == "untiln";

  if(args.size() < 3)
    return;

  reg_t val = strtol(args[args.size()-1].c_str(),NULL,16);
  if(val == LONG_MAX)
    val = strtoul(args[args.size()-1].c_str(),NULL,16);
  std::vector<std::string> args2;
  args2 = std::vector<std::string>(args.begin()+1,args.end()-1);
```

```
  auto func = args[0] == "reg" ? &sim_t::get_reg :
              args[0] == "pc"  ? &sim_t::get_pc :
              args[0] == "mem" ? &sim_t::get_mem :
              NULL;
  if (func == NULL)
    return;
  ctrlc_pressed = false;

  while (1)
  {
    try
    {
      reg_t current = (this->*func)(args2);
      if (cmd_until == (current == val))
        break;
      if (ctrlc_pressed)
        break;
    }
    catch (trap_t& t) {}

    set_procs_debug(noisy);
    step(1);
  }
}
```

添加调试命令

riscv/interactive.cc

第一步：
添加功能代码

```
void sim_t::interactive_<name>() {
  //function code
}
```

第二步：
添加func入口赋值

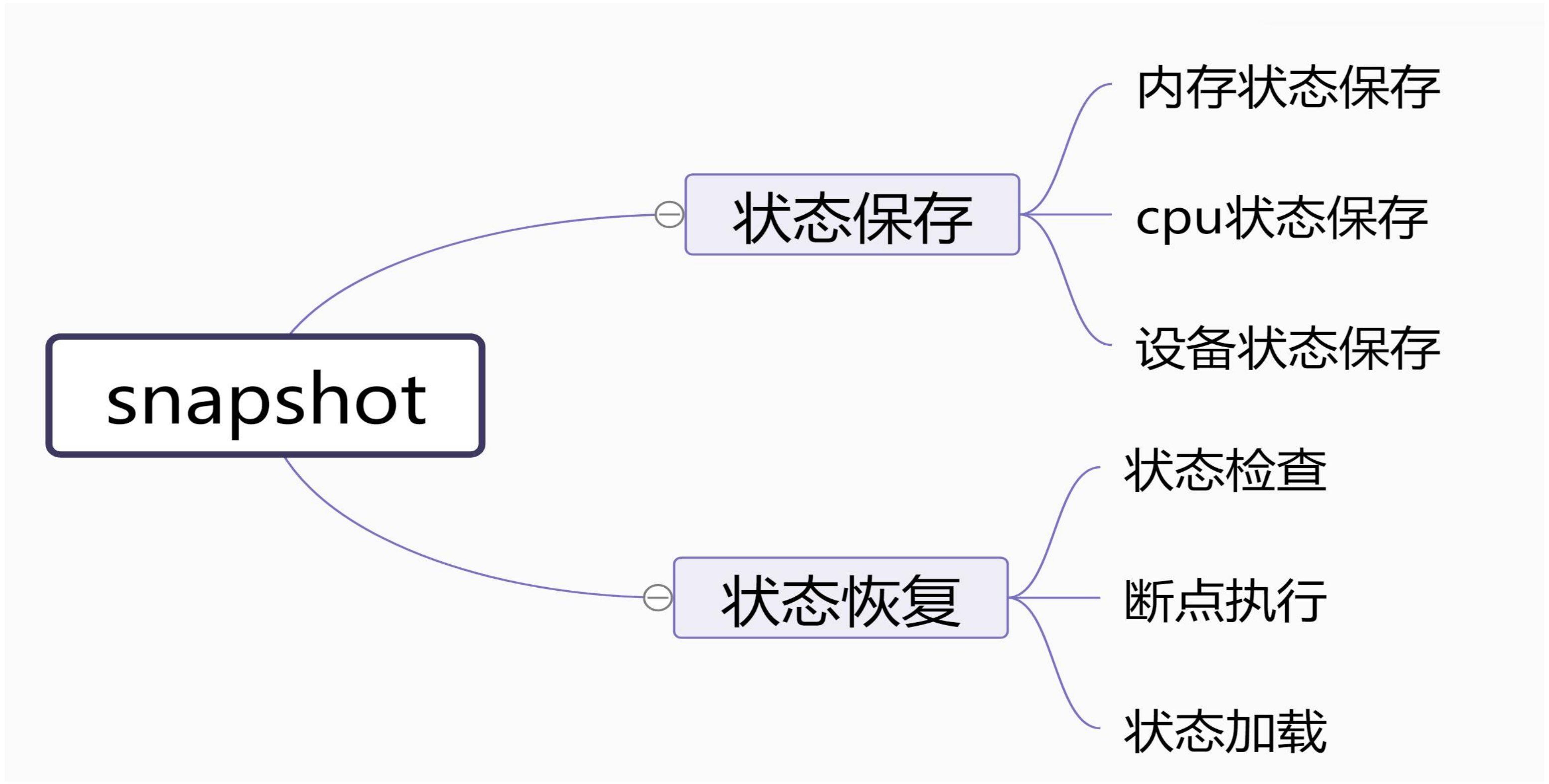
```
void sim_t::interactive()
    funcs["<name>"] = &sim_t::interactive_<name>;
```

第三步：
添加帮助说明

```
void sim_t::interactive_help(const std::string& cmd, const
std::vector<std::string>& args)
```

snapshot机制介绍

snapshot(快照)功能，可以用来保存系统运行到某一时刻的状态，提供某一时刻系统运行状态的完全拷贝，同时支持从该保存点继续运行的能力



github库：https://github.com/plctlab/plct-spike/tree/plct-snapshot

snapshot机制交互调试支持

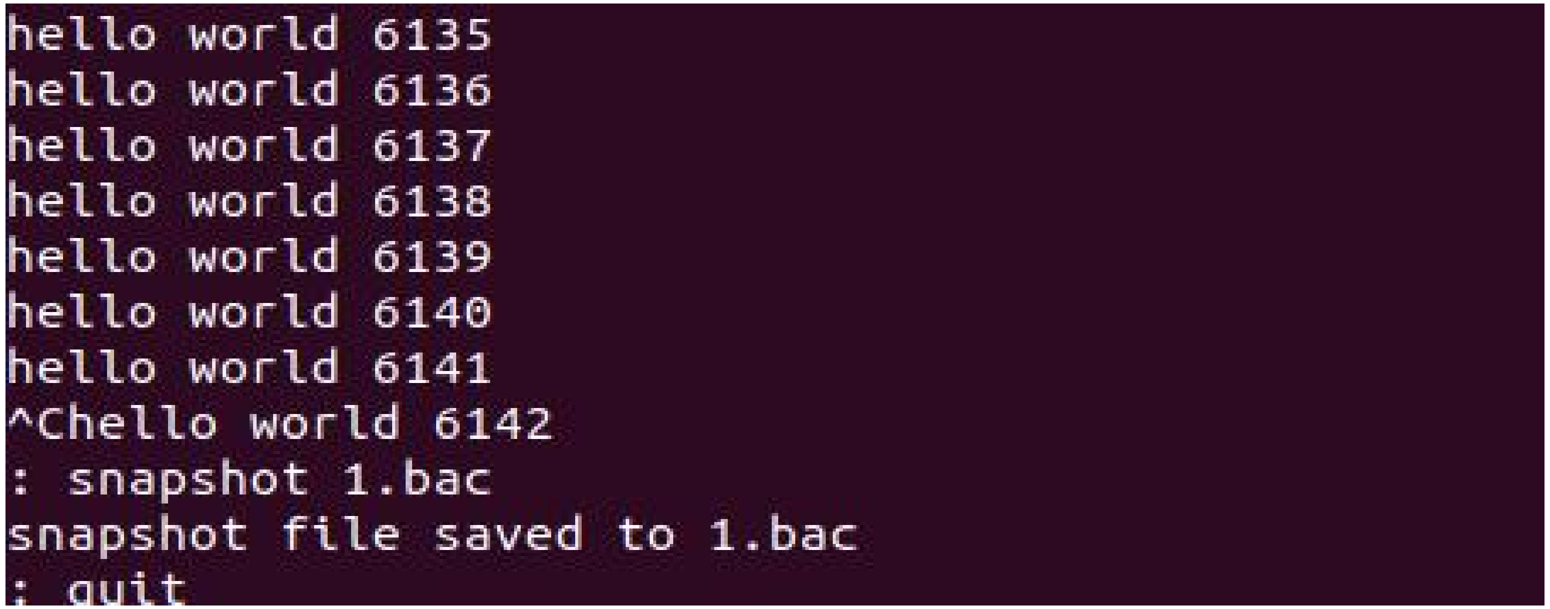
```
void sim_t::interactive_snapshot(const std::string& cmd,
const std::vector<std::string>& args)
{
  if(args.size() < 1)  {
    fprintf(stderr, "illegal argument!\n" );
    return;
  }

  snapshot_t snapshot(this, debug_mmu);
  snapshot.save(args[0].c_str());
  fprintf(stderr, "snapshot file saved to %s\n",args[0].c_str());
}
```

```
void sim_t::interactive()
{
  ...
  funcs["help"] = &sim_t::interactive_help;
  funcs["h"] = funcs["help"];
  funcs["snapshot"] = &sim_t::interactive_snapshot;
  ...
}
```

```
void sim_t::interactive_help(const std::string& cmd, const std::vector<std::string>& args)
{
  std::cerr <<
    ...
    "quit                    # End the simulation\n"
    "snapshot <path>              # save snapshot file to <path>\n"
    ...
}
```

snapshot机制交互调试示例

- snapshot保存：在交互调试模式下通过snapshot命令来保存snapshot文件



```
hello world 6135
hello world 6136
hello world 6137
hello world 6138
hello world 6139
hello world 6140
hello world 6141
^Chello world 6142
: snapshot 1.bac
snapshot file saved to 1.bac
: quit
```

谢谢各位

欢迎提问、讨论、交流合作