中国科学院软件研究所 Institute of Software Chinese Academy of Sciences

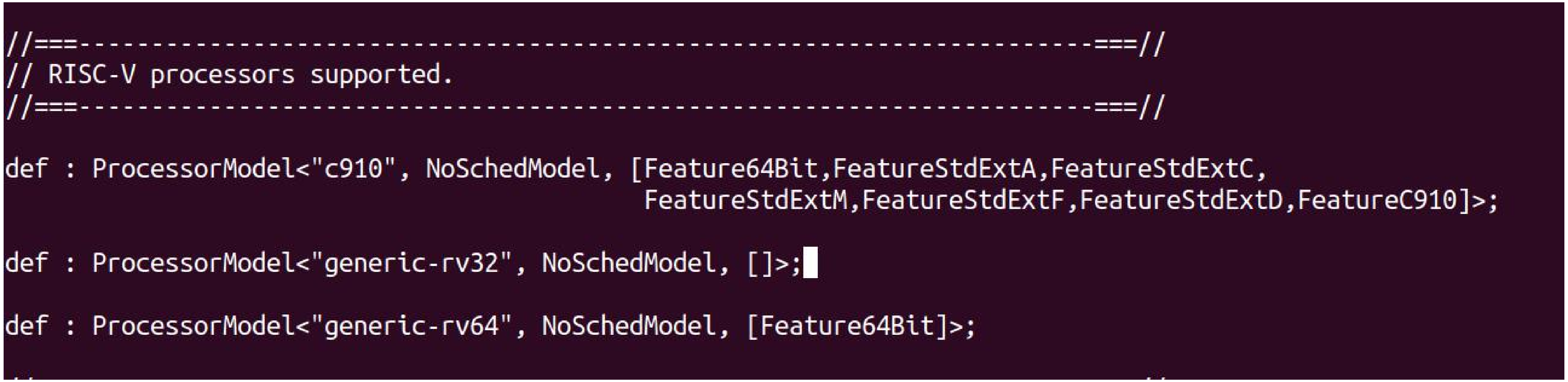
智能软件研究中心 Intelligent Software Research Center

软件所智能软件中心PLCT实验室 王鹏 实习生

2020//02/19

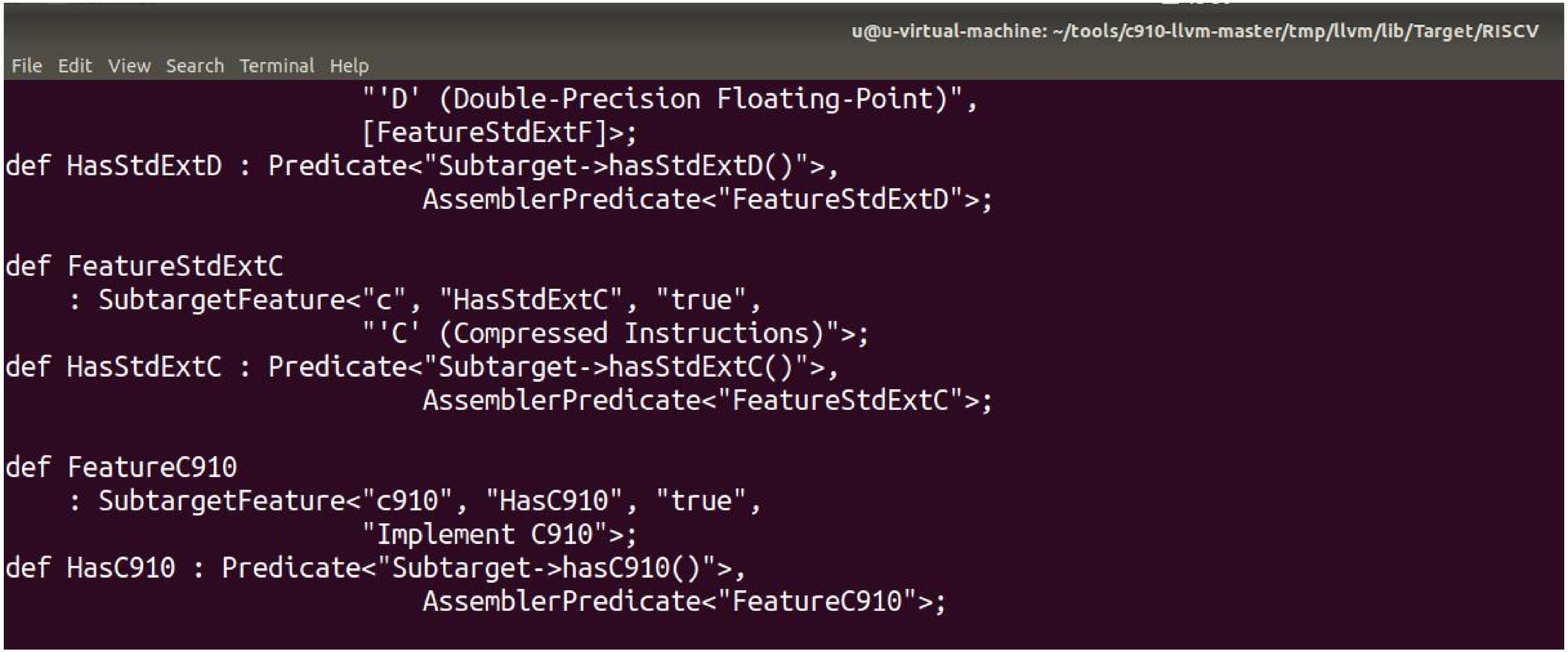
目 录

• 01 玄铁C910 processor in llvm



```
//===----------------------------------------------------------------------===//
// RISC-V processors supported.
//===----------------------------------------------------------------------===//

def : ProcessorModel<"c910", NoSchedModel, [Feature64Bit,FeatureStdExtA,FeatureStdExtC,
                                            FeatureStdExtM,FeatureStdExtF,FeatureStdExtD,FeatureC910]>;

def : ProcessorModel<"generic-rv32", NoSchedModel, []>;

def : ProcessorModel<"generic-rv64", NoSchedModel, [Feature64Bit]>;
```

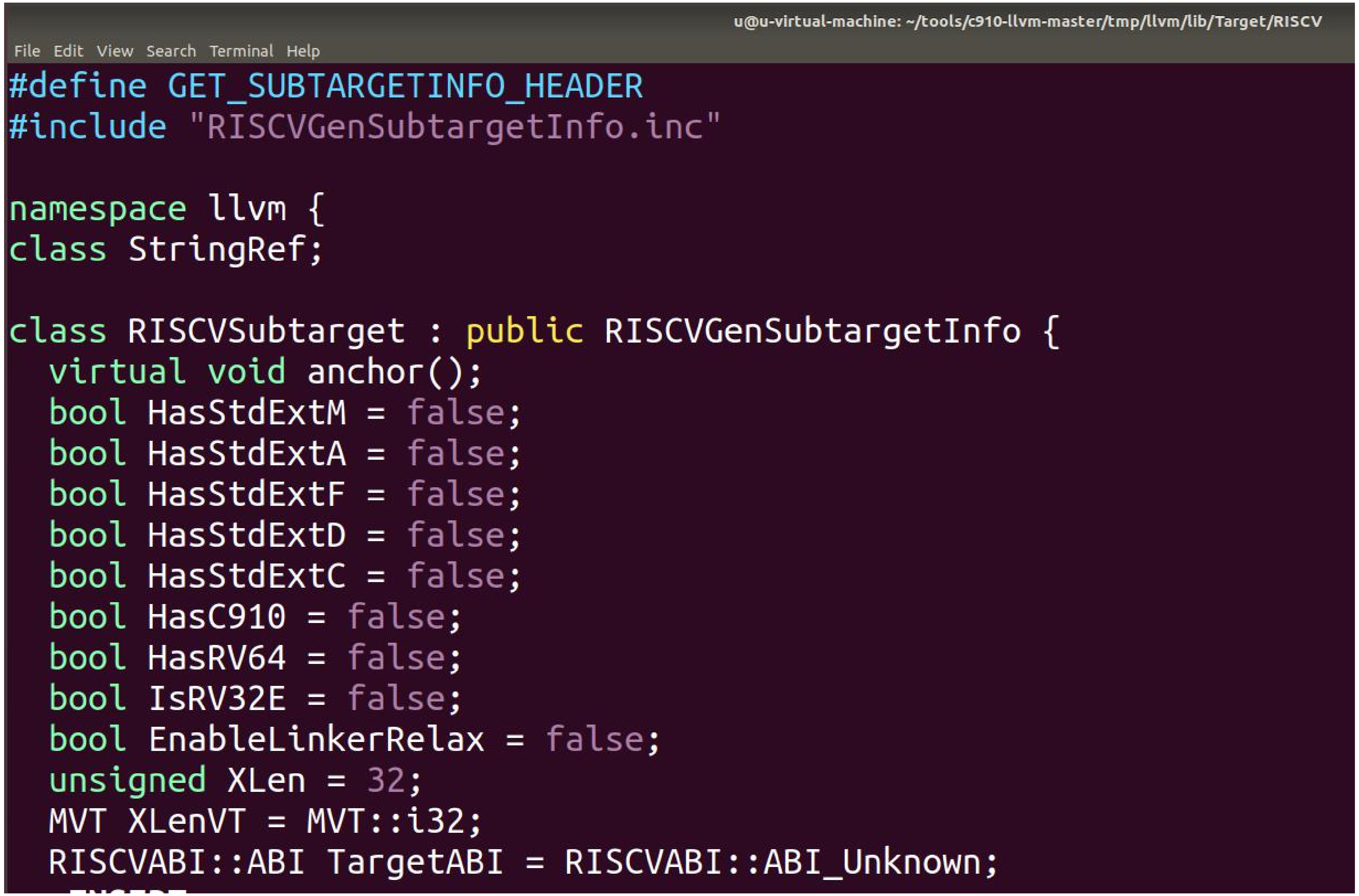
对mcpu=c910在RISCV目录下进行定义


u@u-virtual-machine: ~/tools/c910-llvm-master/tmp/llvm/lib/Target/RISCV
File Edit View Search Terminal Help

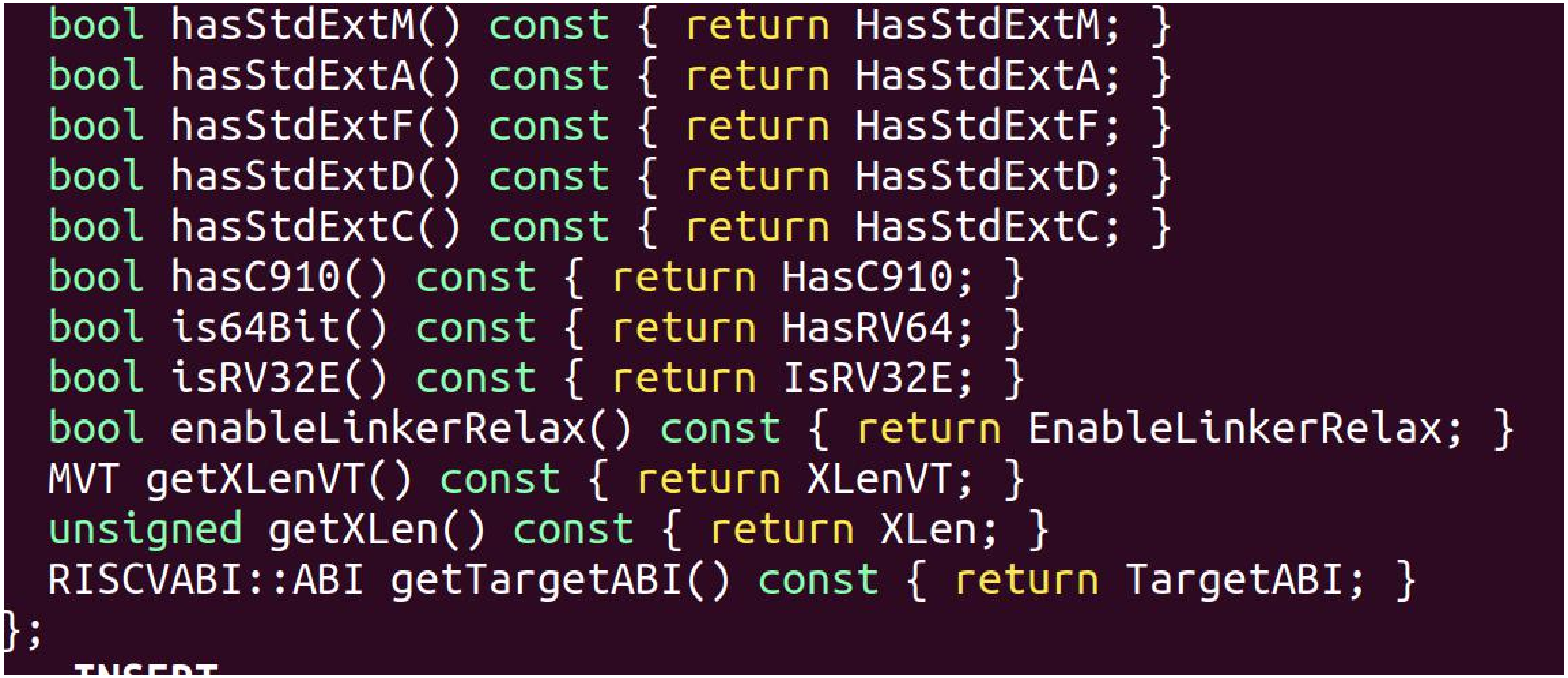
```
                       "'D' (Double-Precision Floating-Point)",
                       [FeatureStdExtF]>;
def HasStdExtD : Predicate<"Subtarget->hasStdExtD()">,
                           AssemblerPredicate<"FeatureStdExtD">;

def FeatureStdExtC
    : SubtargetFeature<"c", "HasStdExtC", "true",
                       "'C' (Compressed Instructions)">;
def HasStdExtC : Predicate<"Subtarget->hasStdExtC()">,
                           AssemblerPredicate<"FeatureStdExtC">;

def FeatureC910
    : SubtargetFeature<"c910", "HasC910", "true",
                       "Implement C910">;
def HasC910 : Predicate<"Subtarget->hasC910()">,
                           AssemblerPredicate<"FeatureC910">;
```

RISCV.td中添加C910

RISCV/RISCVSubtarget.h


u@u-virtual-machine: ~/tools/c910-llvm-master/tmp/llvm/lib/Target/RISCV
File Edit View Search Terminal Help

```
#define GET_SUBTARGETINFO_HEADER
#include "RISCVGenSubtargetInfo.inc"

namespace llvm {
class StringRef;

class RISCVSubtarget : public RISCVGenSubtargetInfo {
  virtual void anchor();
  bool HasStdExtM = false;
  bool HasStdExtA = false;
  bool HasStdExtF = false;
  bool HasStdExtD = false;
  bool HasStdExtC = false;
  bool HasC910 = false;
  bool HasRV64 = false;
  bool IsRV32E = false;
  bool EnableLinkerRelax = false;
  unsigned XLen = 32;
  MVT XLenVT = MVT::i32;
  RISCVABI::ABI TargetABI = RISCVABI::ABI_Unknown;
```



```
  bool hasStdExtM() const { return HasStdExtM; }
  bool hasStdExtA() const { return HasStdExtA; }
  bool hasStdExtF() const { return HasStdExtF; }
  bool hasStdExtD() const { return HasStdExtD; }
  bool hasStdExtC() const { return HasStdExtC; }
  bool hasC910() const { return HasC910; }
  bool is64Bit() const { return HasRV64; }
  bool isRV32E() const { return IsRV32E; }
  bool enableLinkerRelax() const { return EnableLinkerRelax; }
  MVT getXLenVT() const { return XLenVT; }
  unsigned getXLen() const { return XLen; }
  RISCVABI::ABI getTargetABI() const { return TargetABI; }
};
   INSERT
```

• 02 玄铁C910指令集扩展

同步指令子集（4）

Cache指令子集（20）

算术运算指令子集（11）

位操作指令子集（8）

存储指令子集（57）

测试

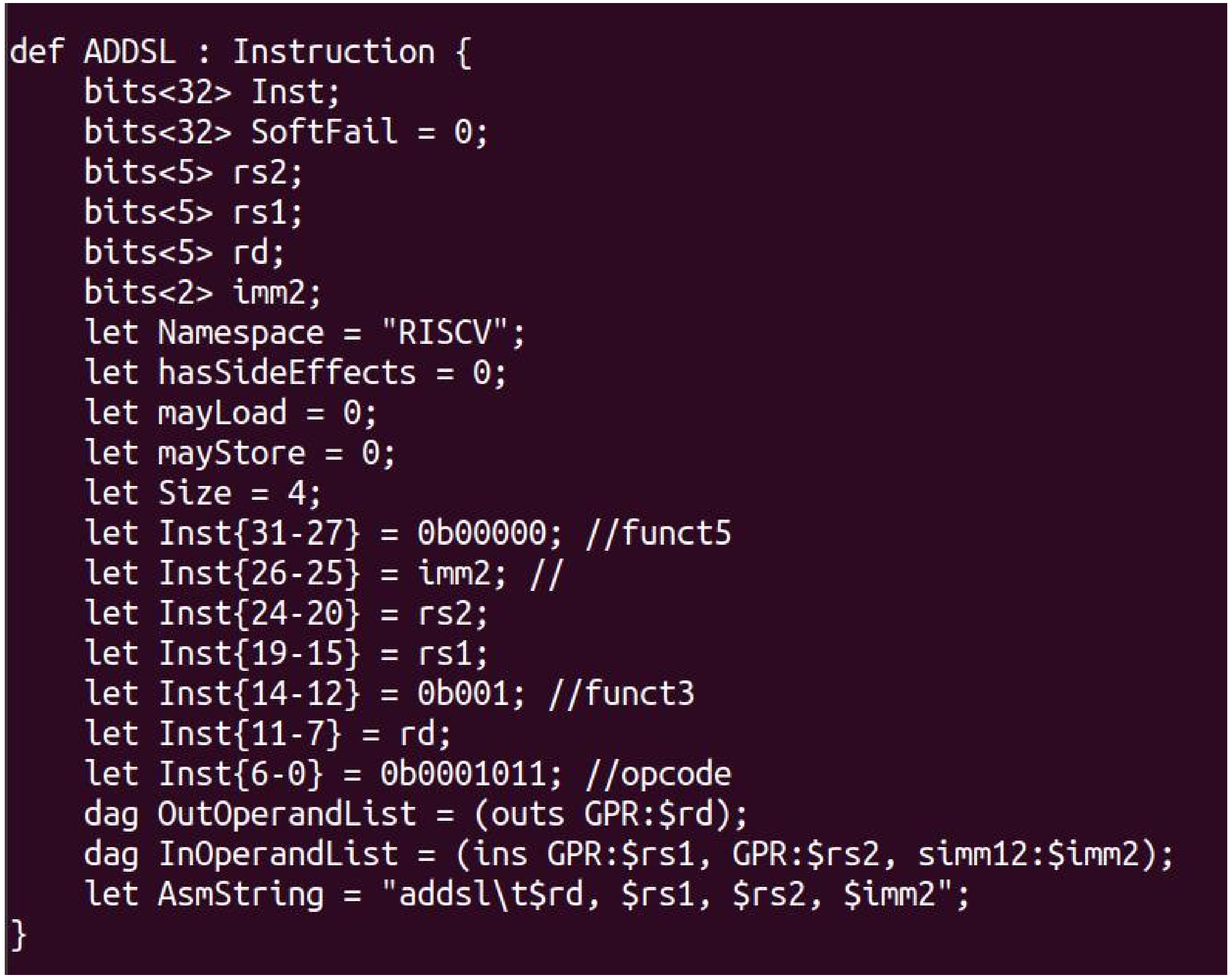
c910-valid.s

6.3.1 ADDSL——寄存器移位相加指令

语法:	addsl rd rs1, rs2, imm2
操作:	rd ← rs1+ rs2<<2
执行权限:	M mode/S mode/U mode
异常:	非法指令异常
指令格式:	

31 27	26 25	24 20	19 15	14 12	11 7	6 0
0 0 0 0 0	imm2	rs2	rs1	0 0 1	rd	0 0 0 1 0 1 1

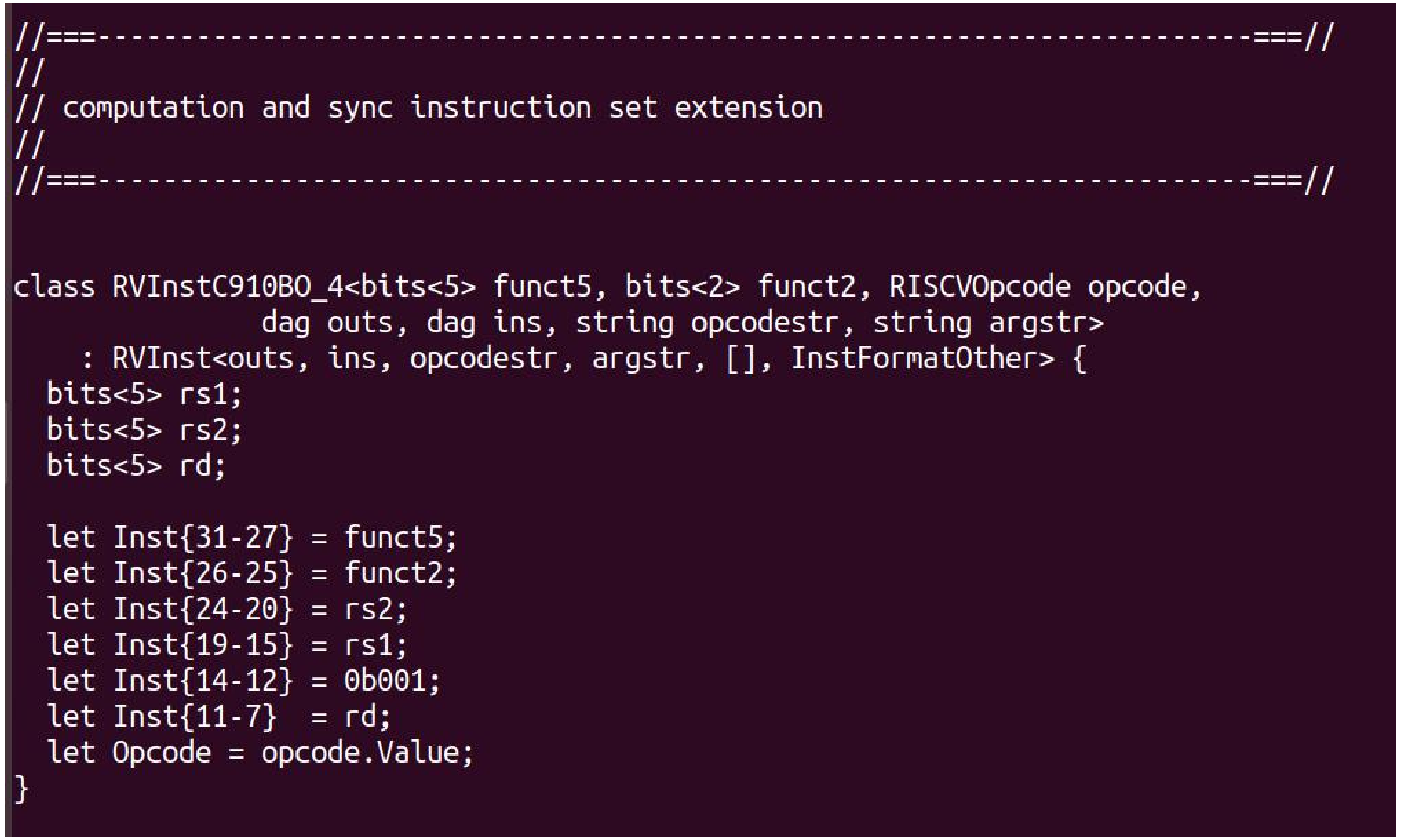
算术运算指令子集



```
def ADDSL : Instruction {
    bits<32> Inst;
    bits<32> SoftFail = 0;
    bits<5> rs2;
    bits<5> rs1;
    bits<5> rd;
    bits<2> imm2;
    let Namespace = "RISCV";
    let hasSideEffects = 0;
    let mayLoad = 0;
    let mayStore = 0;
    let Size = 4;
    let Inst{31-27} = 0b00000; //funct5
    let Inst{26-25} = imm2; //
    let Inst{24-20} = rs2;
    let Inst{19-15} = rs1;
    let Inst{14-12} = 0b001; //funct3
    let Inst{11-7} = rd;
    let Inst{6-0} = 0b0001011; //opcode
    dag OutOperandList = (outs GPR:$rd);
    dag InOperandList = (ins GPR:$rs1, GPR:$rs2, simm12:$imm2);
    let AsmString = "addsl\t$rd, $rs1, $rs2, $imm2";
}
```

算术运算指令子集

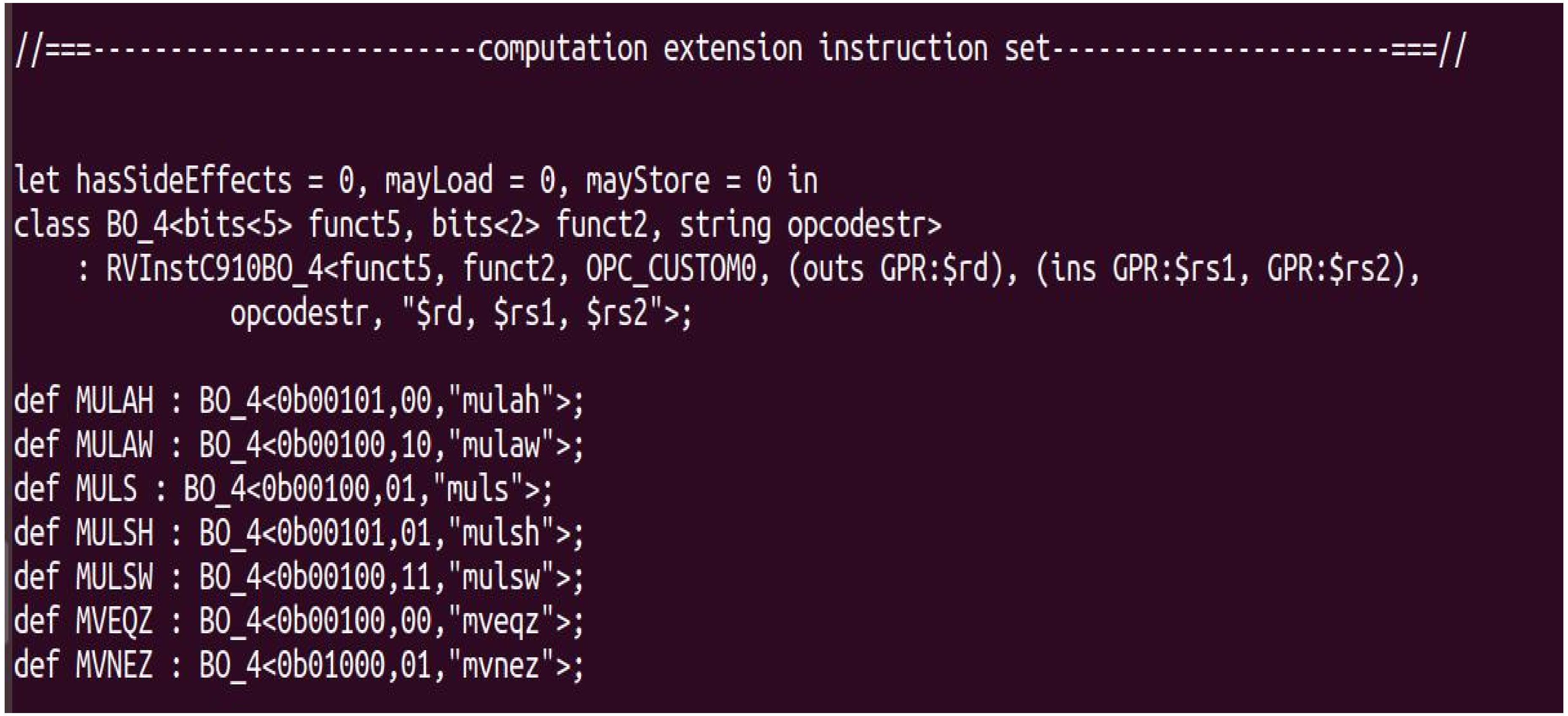
RISCVInstrInfoC910.td



```
//===----------------------------------------------------------------------===//
//
// computation and sync instruction set extension
//
//===----------------------------------------------------------------------===//


class RVInstC910BO_4<bits<5> funct5, bits<2> funct2, RISCVOpcode opcode,
               dag outs, dag ins, string opcodestr, string argstr>
    : RVInst<outs, ins, opcodestr, argstr, [], InstFormatOther> {
  bits<5> rs1;
  bits<5> rs2;
  bits<5> rd;

  let Inst{31-27} = funct5;
  let Inst{26-25} = funct2;
  let Inst{24-20} = rs2;
  let Inst{19-15} = rs1;
  let Inst{14-12} = 0b001;
  let Inst{11-7}  = rd;
  let Opcode = opcode.Value;
}
```

算术运算指令子集

RISCVFormatsC910.td

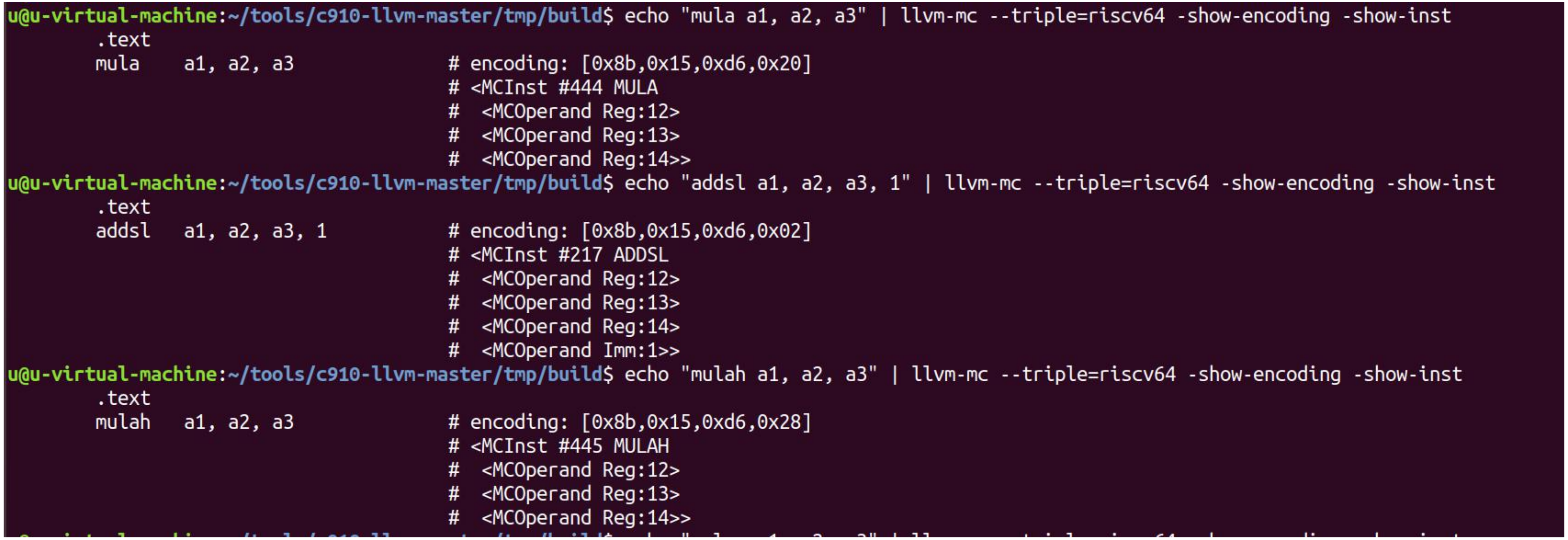


```
//===------------------------computation extension instruction set----------------------===//


let hasSideEffects = 0, mayLoad = 0, mayStore = 0 in
class BO_4<bits<5> funct5, bits<2> funct2, string opcodestr>
    : RVInstC910BO_4<funct5, funct2, OPC_CUSTOM0, (outs GPR:$rd), (ins GPR:$rs1, GPR:$rs2),
              opcodestr, "$rd, $rs1, $rs2">;

def MULAH : BO_4<0b00101,00,"mulah">;
def MULAW : BO_4<0b00100,10,"mulaw">;
def MULS : BO_4<0b00100,01,"muls">;
def MULSH : BO_4<0b00101,01,"mulsh">;
def MULSW : BO_4<0b00100,11,"mulsw">;
def MVEQZ : BO_4<0b00100,00,"mveqz">;
def MVNEZ : BO_4<0b01000,01,"mvnez">;
```

算术运算指令子集

RISCVInstrInfoC910.td

show asm encoding



```
u@u-virtual-machine:~/tools/c910-llvm-master/tmp/build$ echo "mula a1, a2, a3" | llvm-mc --triple=riscv64 -show-encoding -show-inst
        .text
        mula    a1, a2, a3              # encoding: [0x8b,0x15,0xd6,0x20]
                                        # <MCInst #444 MULA
                                        #  <MCOperand Reg:12>
                                        #  <MCOperand Reg:13>
                                        #  <MCOperand Reg:14>>
u@u-virtual-machine:~/tools/c910-llvm-master/tmp/build$ echo "addsl a1, a2, a3, 1" | llvm-mc --triple=riscv64 -show-encoding -show-inst
        .text
        addsl   a1, a2, a3, 1           # encoding: [0x8b,0x15,0xd6,0x02]
                                        # <MCInst #217 ADDSL
                                        #  <MCOperand Reg:12>
                                        #  <MCOperand Reg:13>
                                        #  <MCOperand Reg:14>
                                        #  <MCOperand Imm:1>>
u@u-virtual-machine:~/tools/c910-llvm-master/tmp/build$ echo "mulah a1, a2, a3" | llvm-mc --triple=riscv64 -show-encoding -show-inst
        .text
        mulah   a1, a2, a3              # encoding: [0x8b,0x15,0xd6,0x28]
                                        # <MCInst #445 MULAH
                                        #  <MCOperand Reg:12>
                                        #  <MCOperand Reg:13>
                                        #  <MCOperand Reg:14>>
```

• 03 玄铁C910状态寄存器扩展

机器模式控制寄存器（12）

超级用户模式控制寄存器（3）

用户模式控制寄存器（1）

测试

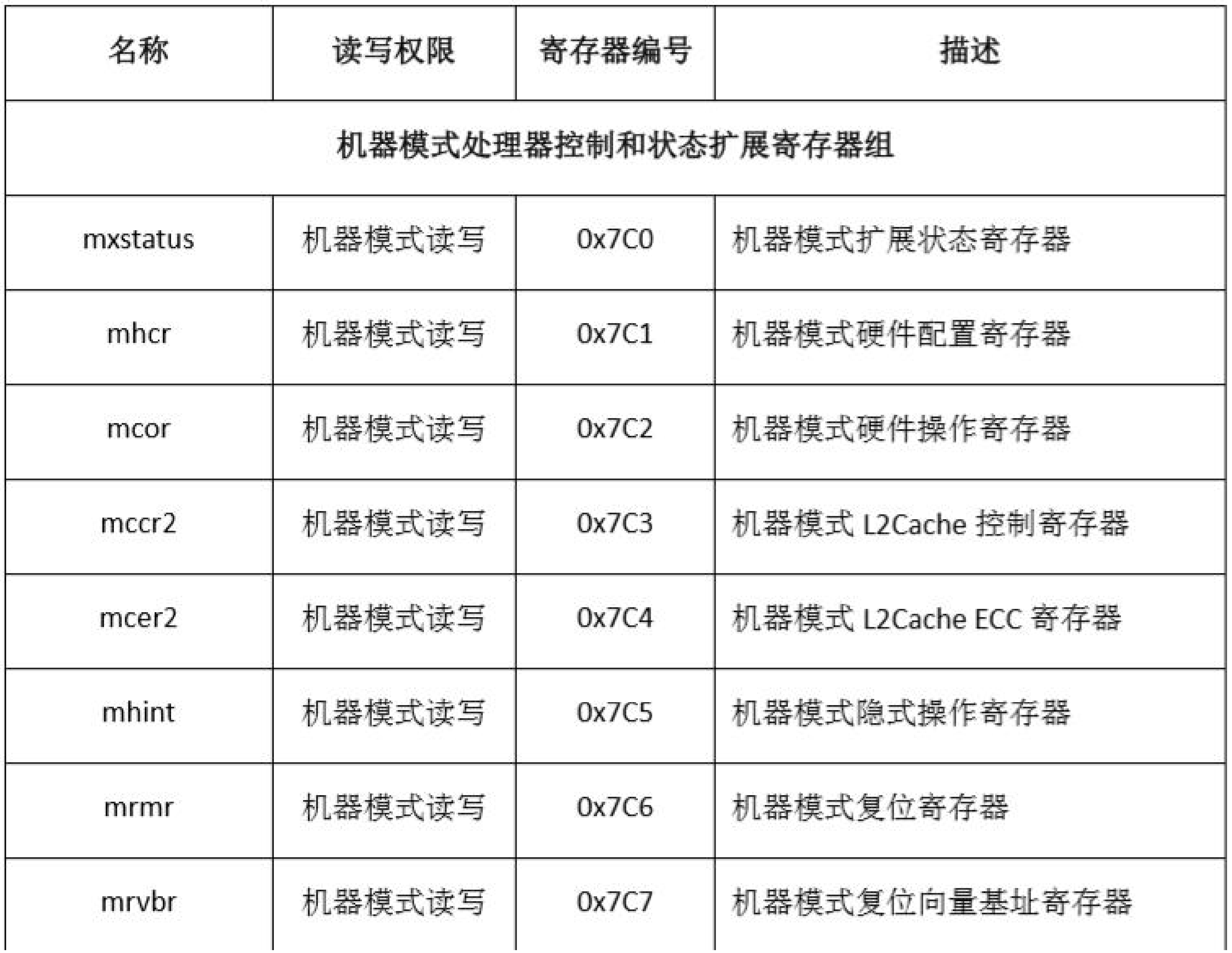
machine-csr-names.s

supervisor-csr-names.s

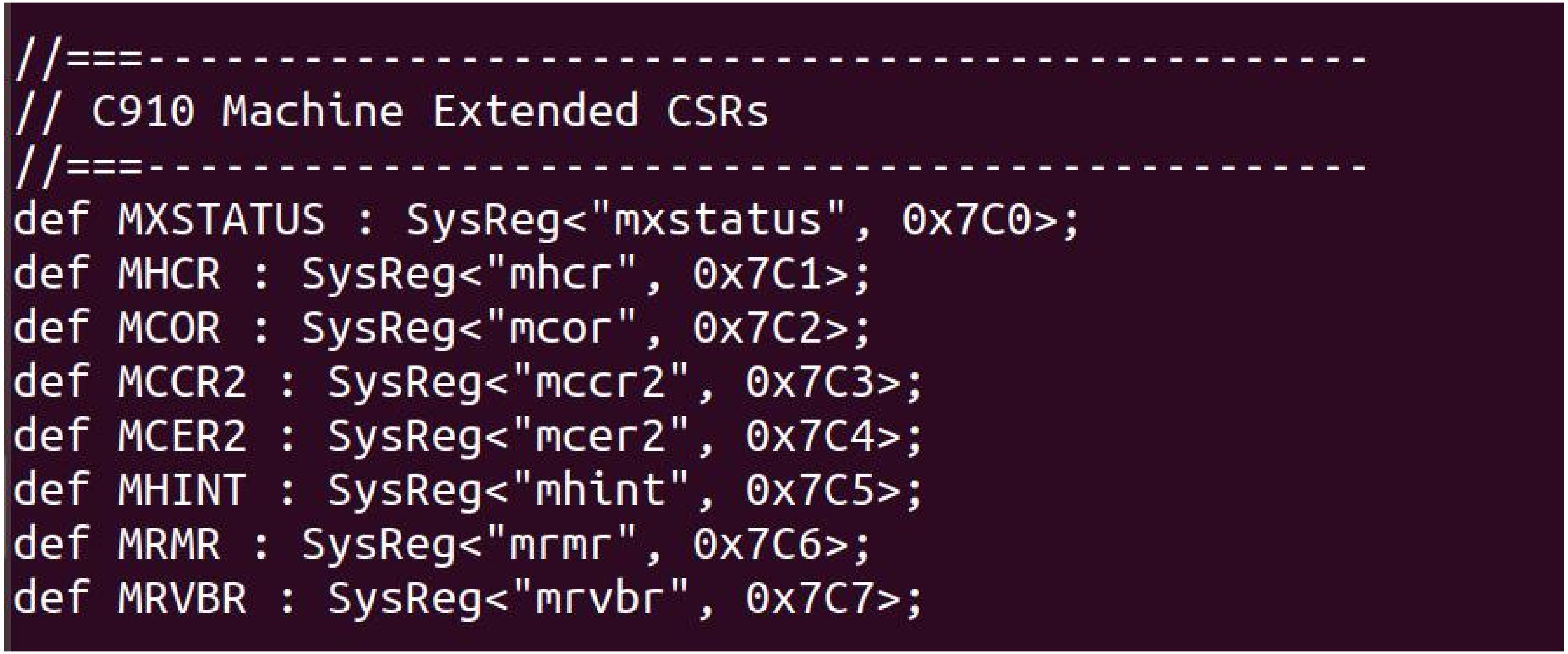
user-csr-names.s

2.5.1.2 玄铁 C910 扩展机器模式控制寄存器

表 2-4 玄铁 C910 扩展机器模式控制寄存器

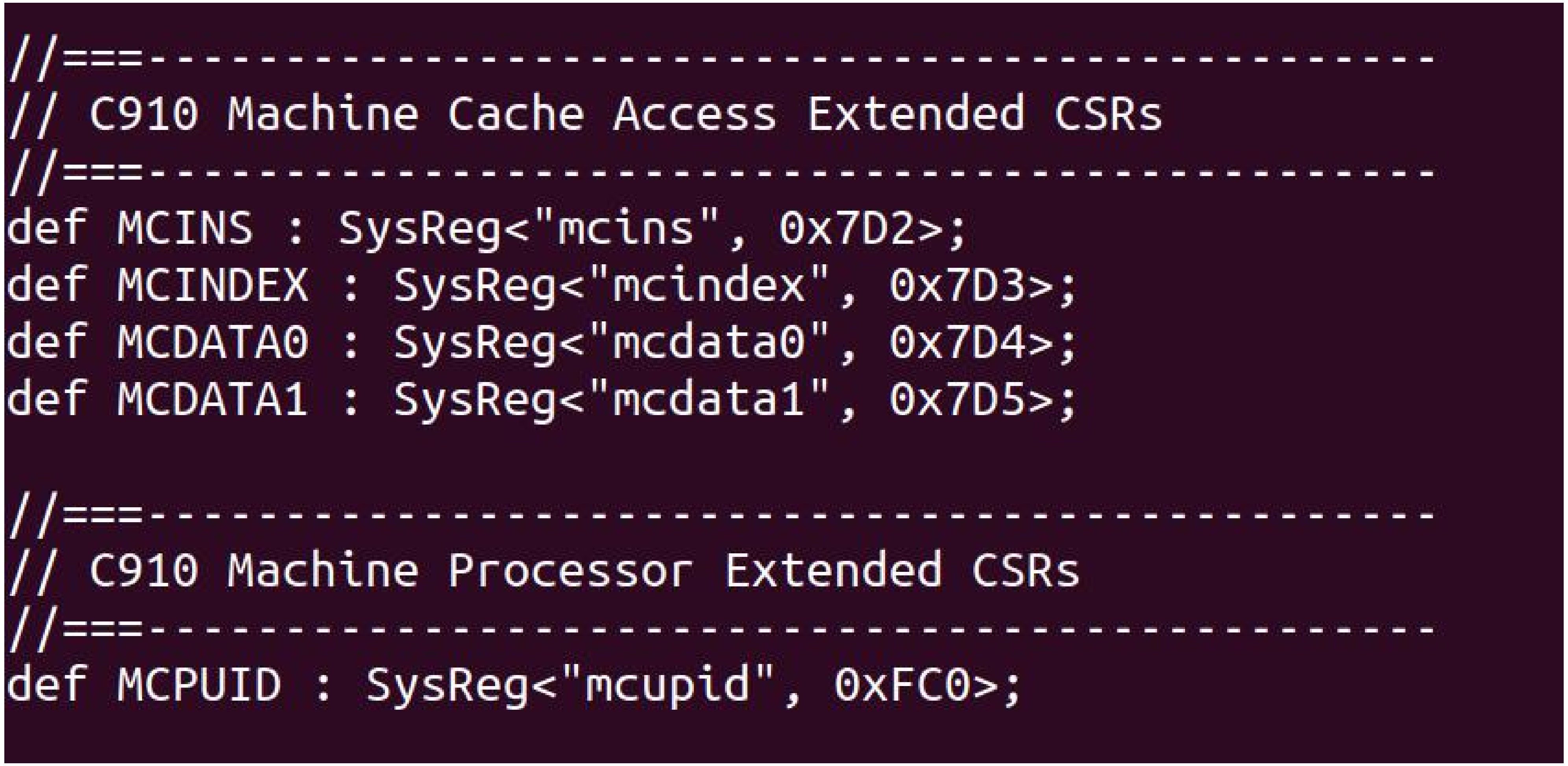


名称	读写权限	寄存器编号	描述
机器模式处理器控制和状态扩展寄存器组			
mxstatus	机器模式读写	0x7C0	机器模式扩展状态寄存器
mhcr	机器模式读写	0x7C1	机器模式硬件配置寄存器
mcor	机器模式读写	0x7C2	机器模式硬件操作寄存器
mccr2	机器模式读写	0x7C3	机器模式 L2Cache 控制寄存器
mcer2	机器模式读写	0x7C4	机器模式 L2Cache ECC 寄存器
mhint	机器模式读写	0x7C5	机器模式隐式操作寄存器
mrmr	机器模式读写	0x7C6	机器模式复位寄存器
mrvbr	机器模式读写	0x7C7	机器模式复位向量基址寄存器



```
//===-----------------------------------------------
// C910 Machine Extended CSRs
//===-----------------------------------------------
def MXSTATUS : SysReg<"mxstatus", 0x7C0>;
def MHCR : SysReg<"mhcr", 0x7C1>;
def MCOR : SysReg<"mcor", 0x7C2>;
def MCCR2 : SysReg<"mccr2", 0x7C3>;
def MCER2 : SysReg<"mcer2", 0x7C4>;
def MHINT : SysReg<"mhint", 0x7C5>;
def MRMR : SysReg<"mrmr", 0x7C6>;
def MRVBR : SysReg<"mrvbr", 0x7C7>;
```

机器模式 Cache 访问扩展寄存器组			
mcins	机器模式读写	0x7D2	机器模式 Cache 指令寄存器
mcindex	机器模式读写	0x7D3	机器模式 Cache 访问索引寄存器
mcdata0	机器模式读写	0x7D4	机器模式 Cache 数据寄存器 0
mcdata1	机器模式读写	0x7D5	机器模式 Cache 数据寄存器 1
机器模式处理器型号扩展寄存器组			
mcupid	机器模式只读	0xFC0	机器模式处理器型号寄存器



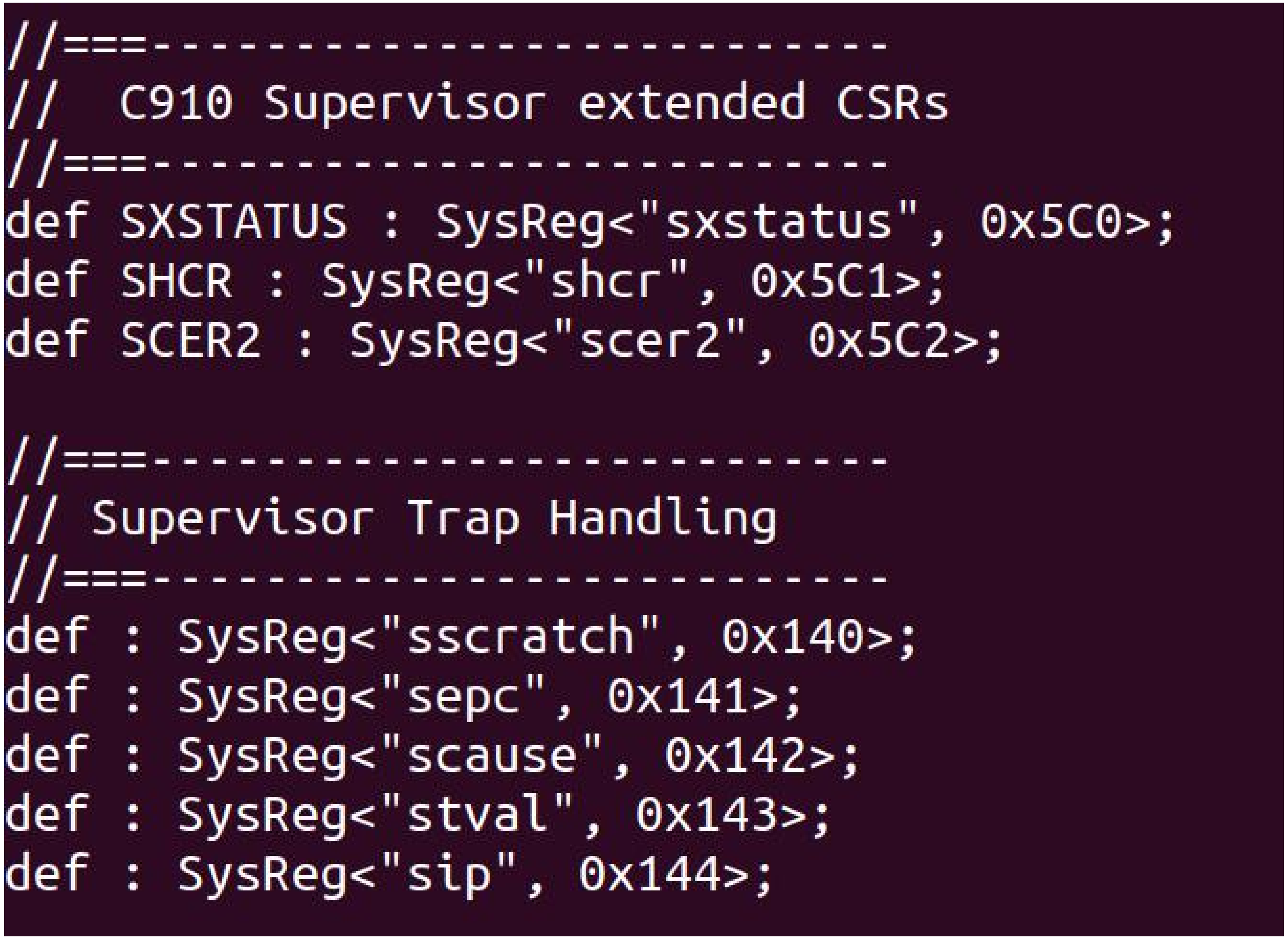
```
//===----------------------------------------------------------
// C910 Machine Cache Access Extended CSRs
//===----------------------------------------------------------
def MCINS : SysReg<"mcins", 0x7D2>;
def MCINDEX : SysReg<"mcindex", 0x7D3>;
def MCDATA0 : SysReg<"mcdata0", 0x7D4>;
def MCDATA1 : SysReg<"mcdata1", 0x7D5>;

//===----------------------------------------------------------
// C910 Machine Processor Extended CSRs
//===----------------------------------------------------------
def MCPUID : SysReg<"mcupid", 0xFC0>;
```

2.5.2.2 玄铁 C910 扩展超级用户模式控制寄存器

表 2-6 为玄铁 C910 中扩展的超级用户模式控制寄存器。

表 2-6 玄铁 C910 扩展超级用户模式控制寄存器

名称	读写权限	寄存器编号	描述
超级用户模式处理器控制和状态扩展寄存器组			
sxstatus	超级用户模式读写	0x5C0	超级用户模式扩展状态寄存器
shcr	超级用户模式只读	0x5C1	超级用户模式硬件配置寄存器
scer2	超级用户模式只读	0x5C2	超级用户模式 L2Cache ECC 寄存器



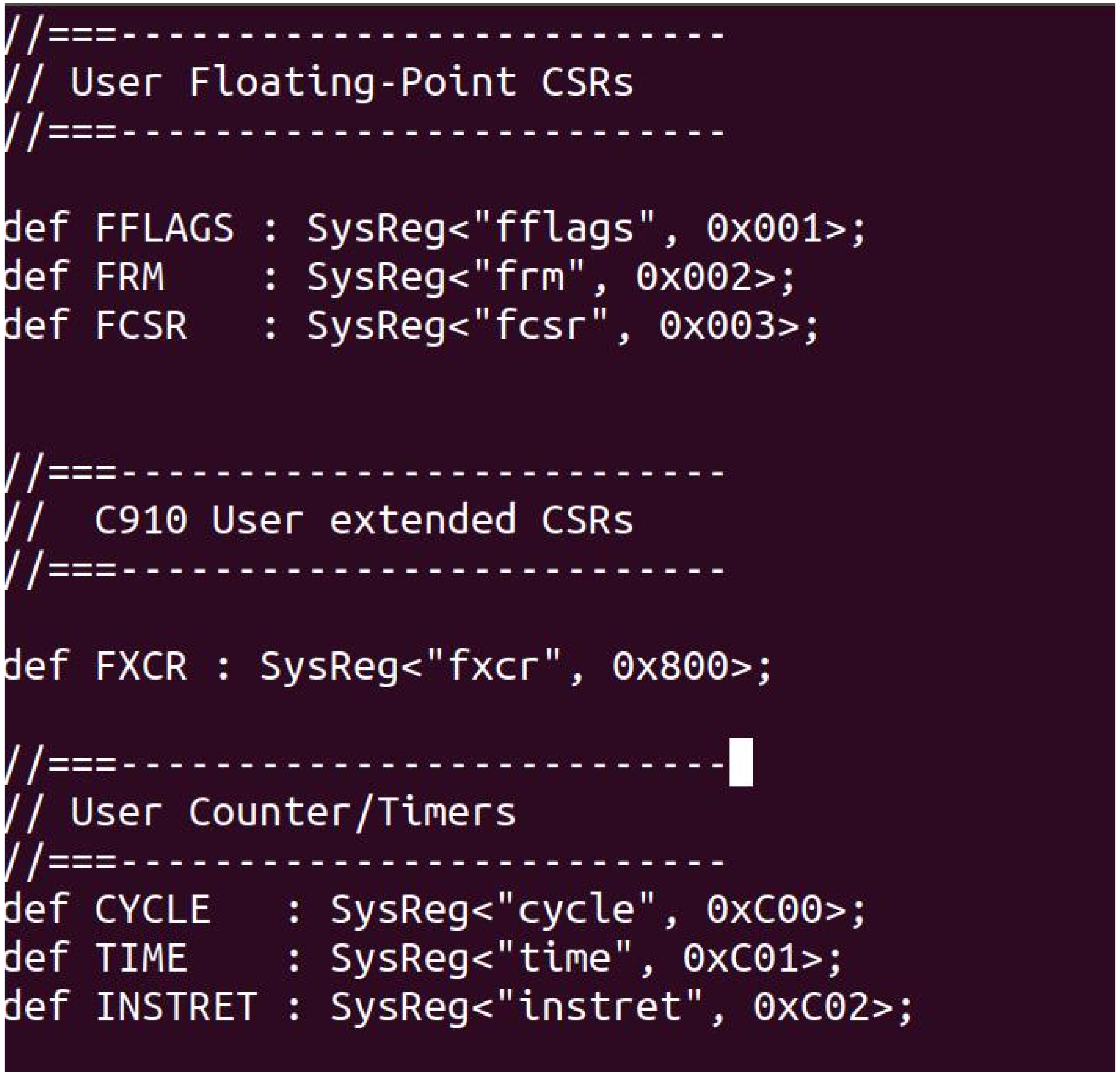
```
//===--------------------------
//  C910 Supervisor extended CSRs
//===--------------------------
def SXSTATUS : SysReg<"sxstatus", 0x5C0>;
def SHCR : SysReg<"shcr", 0x5C1>;
def SCER2 : SysReg<"scer2", 0x5C2>;

//===--------------------------
// Supervisor Trap Handling
//===--------------------------
def : SysReg<"sscratch", 0x140>;
def : SysReg<"sepc", 0x141>;
def : SysReg<"scause", 0x142>;
def : SysReg<"stval", 0x143>;
def : SysReg<"sip", 0x144>;
```

2.5.3.2 玄铁 C910 扩展用户模式控制寄存器

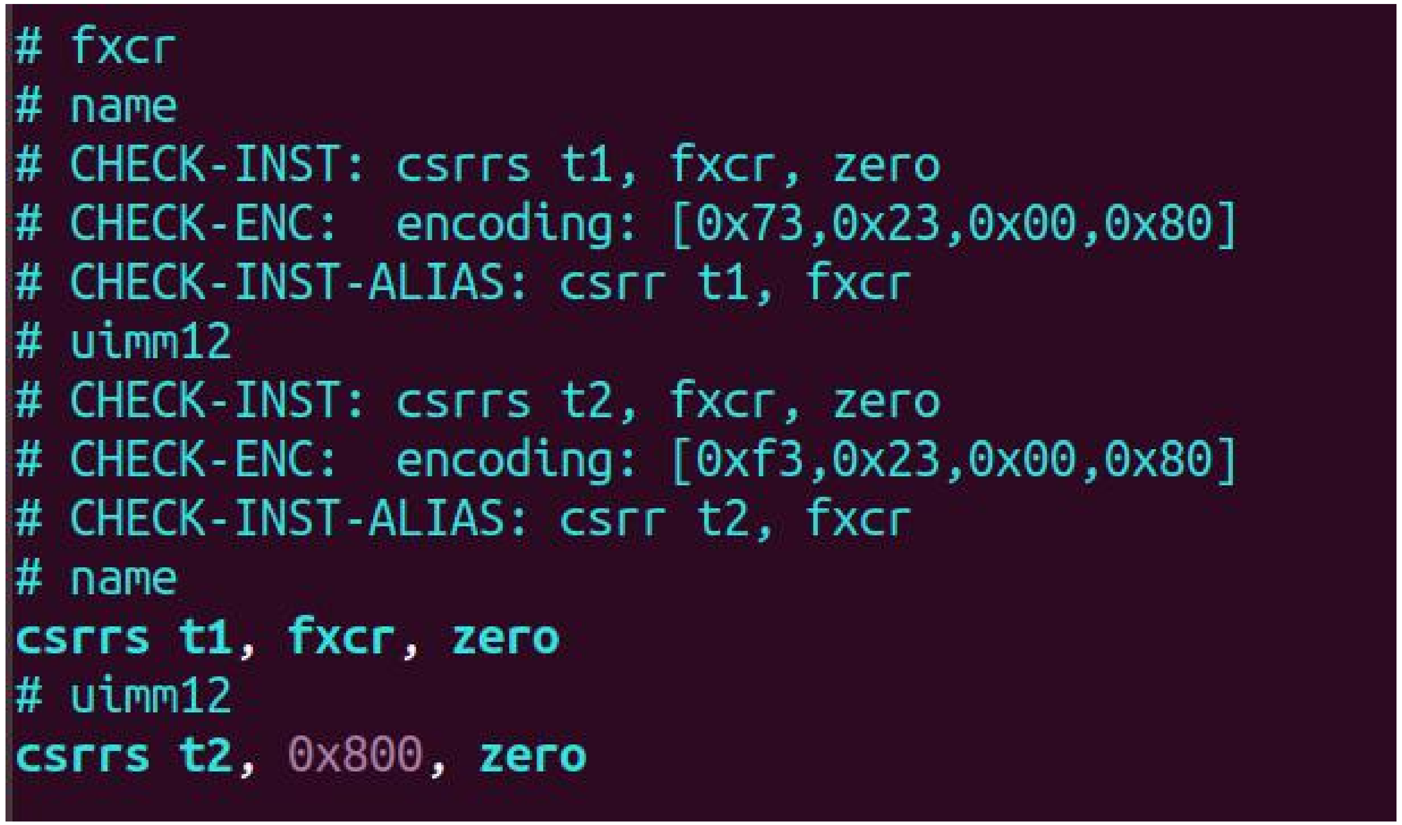
表 2-8 为玄铁 C910 中扩展的用户模式控制寄存器。

表 2-8 玄铁 C910 扩展用户模式控制寄存器

名称	读写权限	寄存器编号	描述
用户模式扩展浮点控制寄存器组			
fxcr	用户模式读写	0x800	用户模式扩展浮点控制寄存器

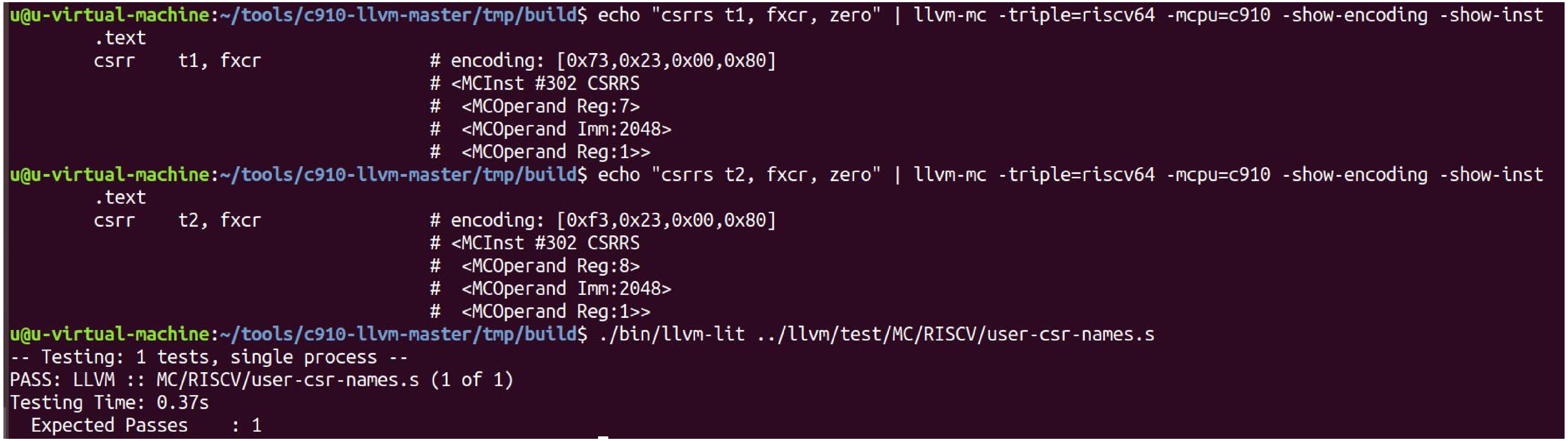


```
//===-----------------------------
// User Floating-Point CSRs
//===-----------------------------

def FFLAGS : SysReg<"fflags", 0x001>;
def FRM    : SysReg<"frm", 0x002>;
def FCSR   : SysReg<"fcsr", 0x003>;


//===-----------------------------
//  C910 User extended CSRs
//===-----------------------------

def FXCR : SysReg<"fxcr", 0x800>;

//===-----------------------------
// User Counter/Timers
//===-----------------------------
def CYCLE   : SysReg<"cycle", 0xC00>;
def TIME    : SysReg<"time", 0xC01>;
def INSTRET : SysReg<"instret", 0xC02>;
```



```
# fxcr
# name
# CHECK-INST: csrrs t1, fxcr, zero
# CHECK-ENC:  encoding: [0x73,0x23,0x00,0x80]
# CHECK-INST-ALIAS: csrr t1, fxcr
# uimm12
# CHECK-INST: csrrs t2, fxcr, zero
# CHECK-ENC:  encoding: [0xf3,0x23,0x00,0x80]
# CHECK-INST-ALIAS: csrr t2, fxcr
# name
csrrs t1, fxcr, zero
# uimm12
csrrs t2, 0x800, zero
```

user-csr-names.s



```
u@u-virtual-machine:~/tools/c910-llvm-master/tmp/build$ echo "csrrs t1, fxcr, zero" | llvm-mc -triple=riscv64 -mcpu=c910 -show-encoding -show-inst
        .text
        csrr    t1, fxcr                        # encoding: [0x73,0x23,0x00,0x80]
                                        # <MCInst #302 CSRRS
                                        #  <MCOperand Reg:7>
                                        #  <MCOperand Imm:2048>
                                        #  <MCOperand Reg:1>>
u@u-virtual-machine:~/tools/c910-llvm-master/tmp/build$ echo "csrrs t2, fxcr, zero" | llvm-mc -triple=riscv64 -mcpu=c910 -show-encoding -show-inst
        .text
        csrr    t2, fxcr                        # encoding: [0xf3,0x23,0x00,0x80]
                                        # <MCInst #302 CSRRS
                                        #  <MCOperand Reg:8>
                                        #  <MCOperand Imm:2048>
                                        #  <MCOperand Reg:1>>
u@u-virtual-machine:~/tools/c910-llvm-master/tmp/build$ ./bin/llvm-lit ../llvm/test/MC/RISCV/user-csr-names.s
-- Testing: 1 tests, single process --
PASS: LLVM :: MC/RISCV/user-csr-names.s (1 of 1)
Testing Time: 0.37s
  Expected Passes    : 1
```

• 04 参考资料

llvm学习手册指导

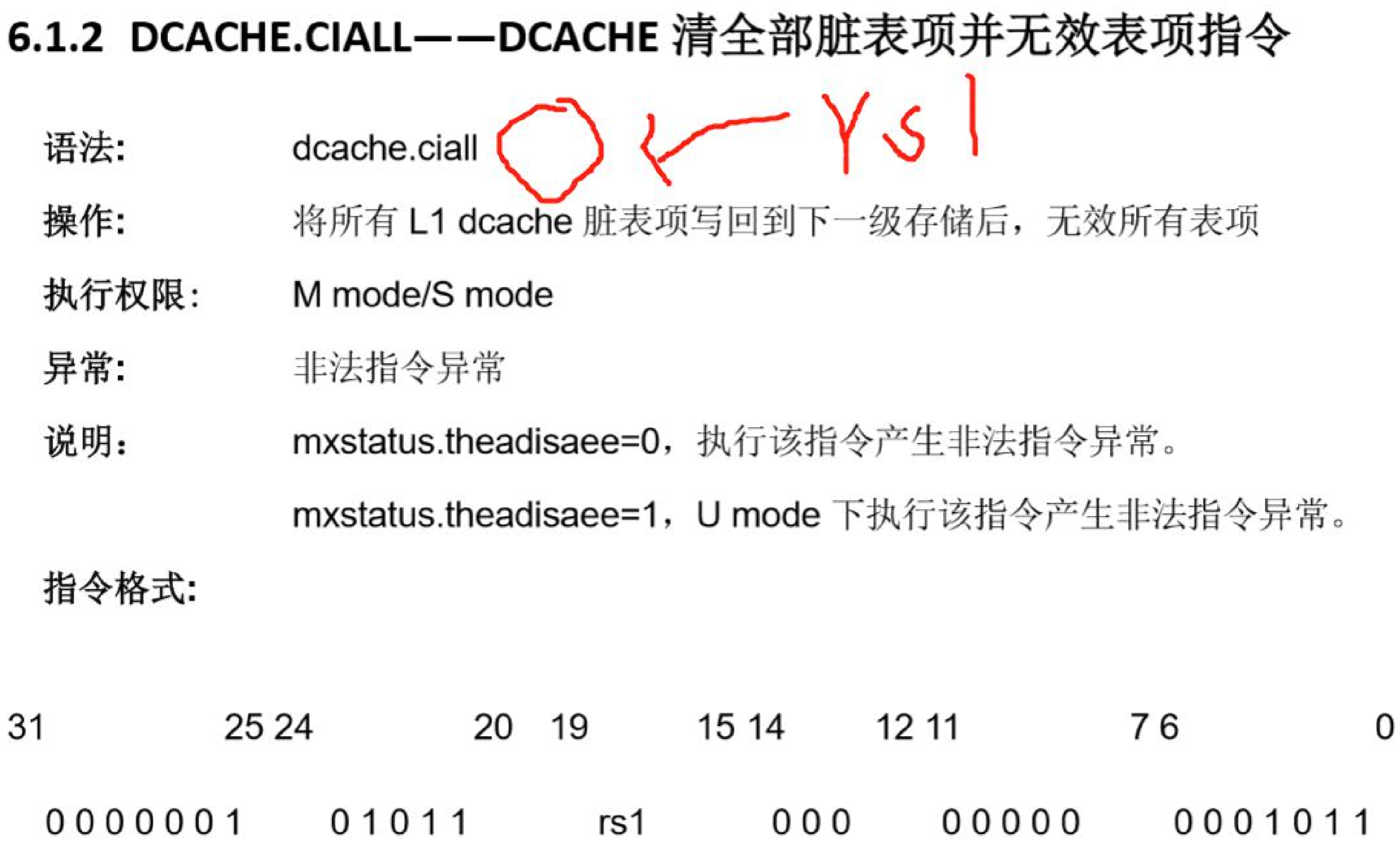
https://github.com/llvm/llvm-project/blob/master/llvm/docs/tutorial

llvm学习笔记（3）

https://blog.csdn.net/wuhui_gdnt/article/details/62884600

《玄铁C910指令集手册》

• 05 勘误



6.1.2 DCACHE.CIALL——DCACHE 清全部脏表项并无效表项指令

语法:	dcache.ciall ○ ← rs1
操作:	将所有 L1 dcache 脏表项写回到下一级存储后，无效所有表项
执行权限:	M mode/S mode
异常:	非法指令异常

说明:	mxstatus.theadisaee=0，执行该指令产生非法指令异常。
	mxstatus.theadisaee=1，U mode 下执行该指令产生非法指令异常。

指令格式:

31　　25	24　　20	19　　15	14　　12	11　　7	6　　0
0 0 0 0 0 0 1	0 1 0 1 1	rs1	0 0 0	0 0 0 0 0	0 0 0 1 0 1 1

谢 谢

欢迎交流合作

2019/02/25