中国科学技术大学
University of Science and Technology of China

计算机组成原理

Lab1 实用汇编程序

计算机实验教学中心

2024/3/18

实验目标

- 理解RISC-V常用32位整数指令功能及编码格式
- 熟悉RISC-V汇编仿真软件RARS，掌握程序调试的基本方法
- 掌握RISC-V简单汇编程序设计，以及存储器初始化文件(COE)的生成方法

实验原理

1.RV32I寄存器：PC和32个通用寄存器

Register	ABI Name	Description
x0	zero	Hard-wired zero 硬编码 0
x1	ra	Return address 返回地址
x2	sp	Stack pointer 栈指针
x3	gp	Global pointer 全局指针
x4	tp	Thread pointer 线程指针
x5	t0	Temporary/alternate link register
x6–7	t1–2	Temporaries 临时寄存器
x8	s0/fp	Saved register/frame pointer
x9	s1	Saved register 保存寄存器
x10–11	a0–1	Function arguments/return values
x12–17	a2–7	Function arguments 函数参数
x18–27	s2–11	Saved registers 保存寄存器
x28–31	t3–6	Temporaries 临时寄存器

RV32I寄存器汇编助记符

实验原理

2.RV32I指令类型

□ 运算类

- ✓ 算术：add, sub, addi, auipc, lui
- ✓ 逻辑：and, or, xor, andi, ori, xori
- ✓ 移位(shift)：sll, srl, sra, slli, srli, srai
- ✓ 比较(set if less than)：slt, sltu, slti, sltiu

Category	*Name*	*Fmt*	*RV32I Base*	
Shifts	Shift Left Logical	R	SLL	rd,rs1,rs2
	Shift Left Log. Imm.	I	SLLI	rd,rs1,shamt
	Shift Right Logical	R	SRL	rd,rs1,rs2
	Shift Right Log. Imm.	I	SRLI	rd,rs1,shamt
	Shift Right Arithmetic	R	SRA	rd,rs1,rs2
	Shift Right Arith. Imm.	I	SRAI	rd,rs1,shamt
Arithmetic	ADD	R	ADD	rd,rs1,rs2
	ADD Immediate	I	ADDI	rd,rs1,imm
	SUBtract	R	SUB	rd,rs1,rs2
	Load Upper Imm	U	LUI	rd,imm
	Add Upper Imm to PC	U	AUIPC	rd,imm
Logical	XOR	R	XOR	rd,rs1,rs2
	XOR Immediate	I	XORI	rd,rs1,imm
	OR	R	OR	rd,rs1,rs2
	OR Immediate	I	ORI	rd,rs1,imm
	AND	R	AND	rd,rs1,rs2
	AND Immediate	I	ANDI	rd,rs1,imm
Compare	Set <	R	SLT	rd,rs1,rs2
	Set < Immediate	I	SLTI	rd,rs1,imm
	Set < Unsigned	R	SLTU	rd,rs1,rs2
	Set < Imm Unsigned	I	SLTIU	rd,rs1,imm

实验原理

2.RV32I指令类型

- 访存类
  - ✓ 加载(load)：lw
  - ✓ 存储(store)：sw
- 转移类
  - ✓ 分支(branch)：beq, blt, bltu, bne, bge, bgeu
  - ✓ 跳转(jump)：jal, jalr

Category	Name	Fmt	RV32I Base	
Branches	Branch =	B	BEQ	rs1,rs2,imm
	Branch ≠	B	BNE	rs1,rs2,imm
	Branch <	B	BLT	rs1,rs2,imm
	Branch ≥	B	BGE	rs1,rs2,imm
	Branch < Unsigned	B	BLTU	rs1,rs2,imm
	Branch ≥ Unsigned	B	BGEU	rs1,rs2,imm
Jump & Link	J&L	J	JAL	rd,imm
	Jump & Link Register	I	JALR	rd,rs1,imm
Loads	Load Byte	I	LB	rd,rs1,imm
	Load Halfword	I	LH	rd,rs1,imm
	Load Byte Unsigned	I	LBU	rd,rs1,imm
	Load Half Unsigned	I	LHU	rd,rs1,imm
	Load Word	I	LW	rd,rs1,imm
Stores	Store Byte	S	SB	rs1,rs2,imm
	Store Halfword	S	SH	rs1,rs2,imm
	Store Word	S	SW	rs1,rs2,imm

实验原理

3.RV32I指令格式及功能

- 指令长度固定32位，不同指令使用不同指令格式
- 尽量保持不同指令中相同字段的位置，降低硬件复杂性

31	30 25	24 21	20	19 15	14 12	11 8	7	6 0	
funct7		rs2		rs1	funct3	rd		opcode	R-type
imm[11:0]				rs1	funct3	rd		opcode	I-type
imm[11:5]		rs2		rs1	funct3	imm[4:0]		opcode	S-type
imm[12]	imm[10:5]	rs2		rs1	funct3	imm[4:1]	imm[11]	opcode	B-type
imm[31:12]						rd		opcode	U-type
imm[20]	imm[10:1]		imm[11]	imm[19:12]		rd		opcode	J-type

实验原理

3.RV32I指令格式及功能

□ 运算指令

✓ **add rd, rs1, rs2**　　　　**# x[rd] = x[rs1] + x[rs2]**

31 25	24 20	19 15	14 12	11 7	6 0
funct7	rs2	rs1	funct3	rd	opcode
7	5	5	3	5	7
0000000	src2	src1	ADD/SLT/SLTU	dest	OP
0000000	src2	src1	AND/OR/XOR	dest	OP
0000000	src2	src1	SLL/SRL	dest	OP
0100000	src2	src1	SUB/SRA	dest	OP

✓ **addi rd, rs1, imm**　　　　**# x[rd] = x[rs1] + sext(imm)**

31 20	19 15	14 12	11 7	6 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
I-immediate[11:0]	src	ADDI/SLTI[U]	dest	OP-IMM
I-immediate[11:0]	src	ANDI/ORI/XORI	dest	OP-IMM

实验原理

3.RV32I指令格式及功能

□ 运算指令

- ✓ lui rd, imm   # x[rd] = sext(imm[31:12] << 12)
- ✓ auipc rd, imm   # x[rd] = pc + sext(imm[31:12] << 12)

31 … 12	11 … 7	6 … 0
imm[31:12]	rd	opcode
20	5	7
U-immediate[31:12]	dest	LUI
U-immediate[31:12]	dest	AUIPC

实验原理

3.RV32I指令格式及功能

◻ 访存指令

✓ **lw rd, offset(rs1)** # x[rd] = M[x[rs1] + sext(offset)]

31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
offset[11:0]	base	width	dest	LOAD

✓ **sw rs2, offset(rs1)** # M[x[rs1]+sext(offset)=x[rs2]

31 — 25	24 — 20	19 — 15	14 — 12	11 — 7	6 — 0
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
7	5	5	3	5	7
offset[11:5]	src	base	width	offset[4:0]	STORE

实验原理

3.RV32I指令格式及功能

□ 分支指令

- ✓ beq rs1, rs2, offset # if (rs1 == rs2) pc += sext(offset)
- ✓ blt rs1, rs2, offset # if (rs1 < rs2) pc += sext(offset)

31	30 25	24 20	19 15	14 12	11 8	7	6 0
imm[12]	imm[10:5]	rs2	rs1	funct3	imm[4:1]	imm[11]	opcode
1	6	5	5	3	4	1	7
offset[12,10:5]		src2	src1	BEQ/BNE	offset[11,4:1]		BRANCH
offset[12,10:5]		src2	src1	BLT[U]	offset[11,4:1]		BRANCH
offset[12,10:5]		src2	src1	BGE[U]	offset[11,4:1]		BRANCH

实验原理

3.RV32I指令格式及功能

□ 跳转指令

✓ **jal rd, offset** # x[rd] = pc+4; pc += sext(offset)

31	30 … 21	20	19 … 12	11 … 7	6 … 0
imm[20]	imm[10:1]	imm[11]	imm[19:12]	rd	opcode
1	10	1	8	5	7
offset[20:1]				dest	JAL

✓ **jalr rd, offset(rs1)** # t =pc+4; pc=(x[rs1]+sext(offset))&~1; x[rd]=t

31 … 20	19 … 15	14 … 12	11 … 7	6 … 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
offset[11:0]	base	0	dest	JALR

实验原理

4.汇编指示符和伪指令

- **汇编指示符（Assembly Directives)**

  .data, .text

  .word, .half, .byte, .string

  .align ……

- **伪指令（Pseudo Instructions)**

  li, la, mv

  nop, not, neg

  j, jr, call, ret ……

```
Example:
.eqv CONSTANT, 0xdeadbeef
.data
        myarray: .word 1  2
.text
li a0, CONSTANT

# lui a0,0xdeadc
# addi a0,a0,0xffffeef
```

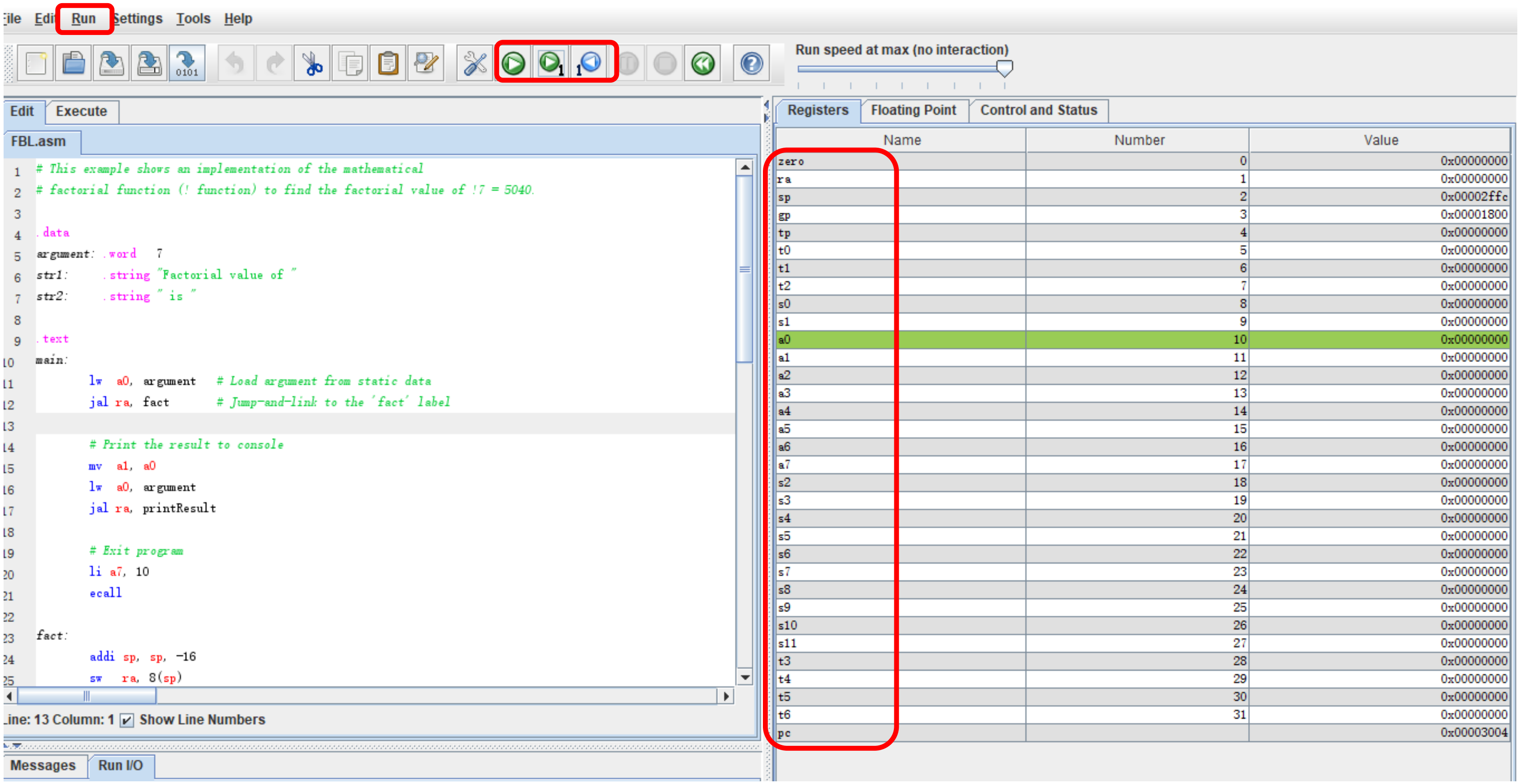
- **参考资料：RISC-V Assembly Programmer's Manual**

  https://github.com/riscv-non-isa/riscv-asm-manual/blob/master/riscv-asm.md#risc-v-assembly-programmers-manual

实验原理

5.RARS:RISC-V Assembler & Runtime Simulator

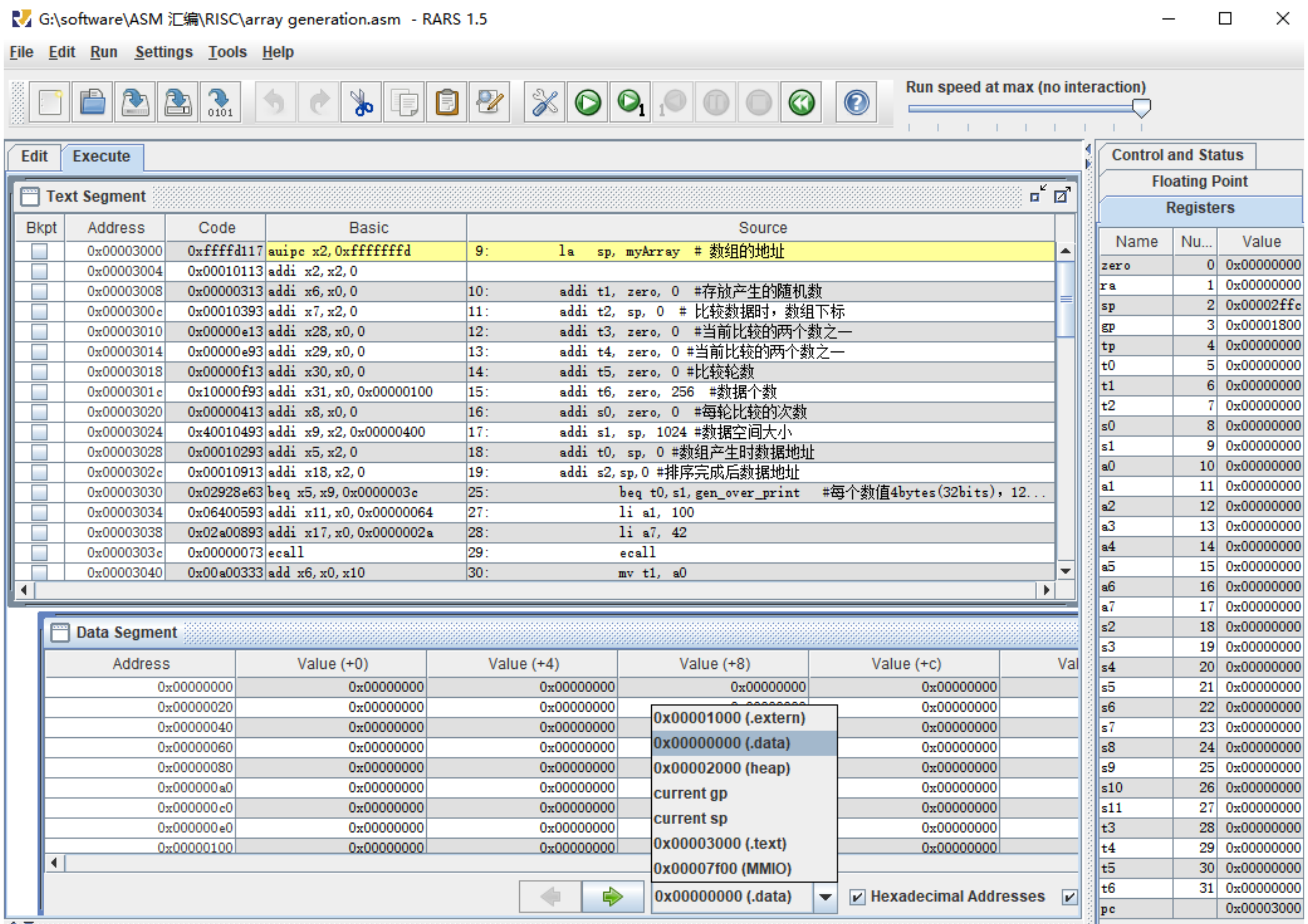
□ 界面介绍



实验原理

5.RARS:RISC-V Assembler & Runtime Simulator

□ 界面介绍



实验原理

5.RARS:RISC-V Assembler & Runtime Simulator

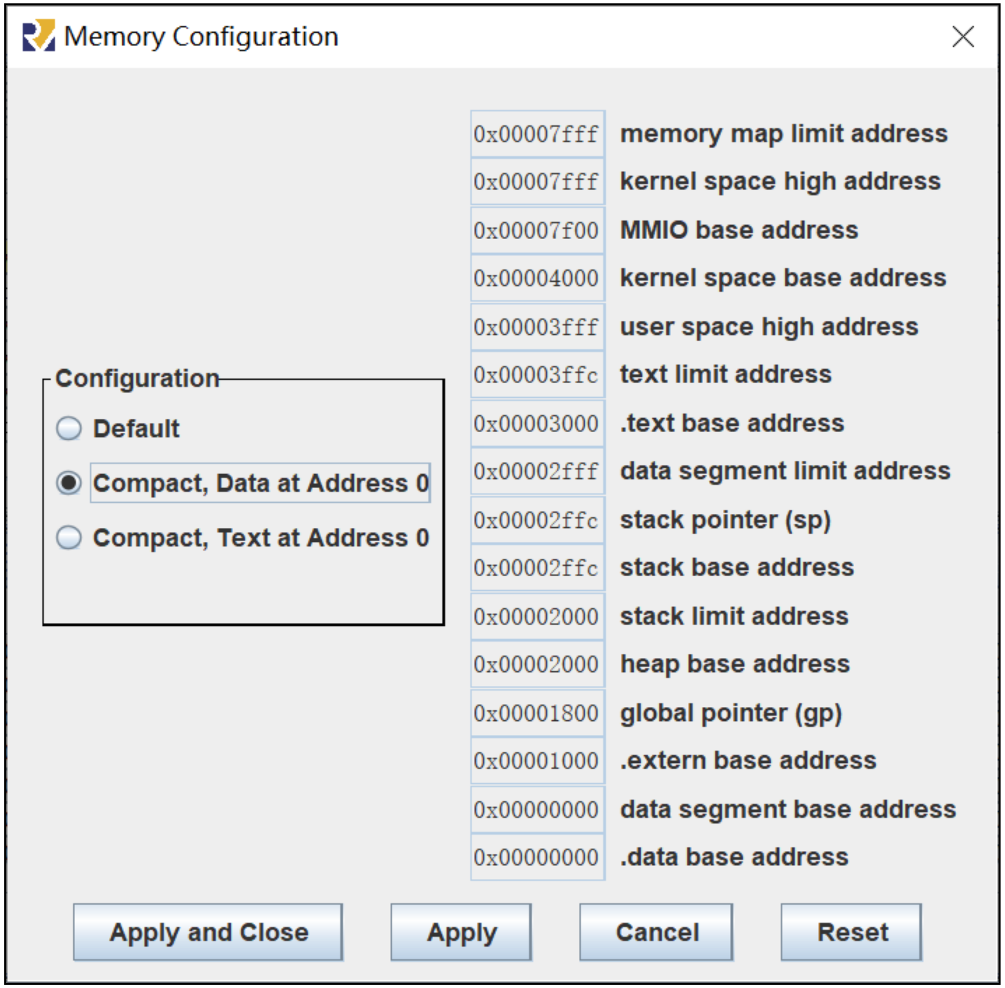
- **存储器配置**
- ✓ **Setting >> Memory Configuration**
- ✓ **假定配置为紧凑型**

数据地址：

– 0x0000 ~ 0x2fff

代码地址：

– 0x3000 ~ 0x3fff

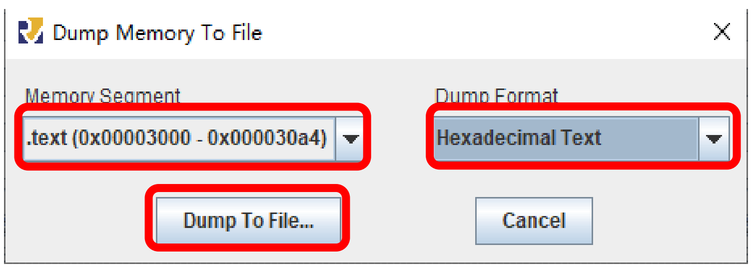


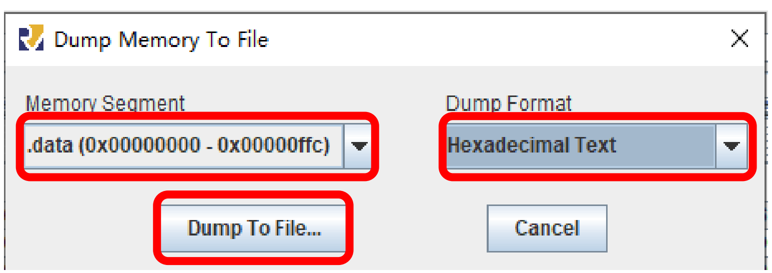
实验原理

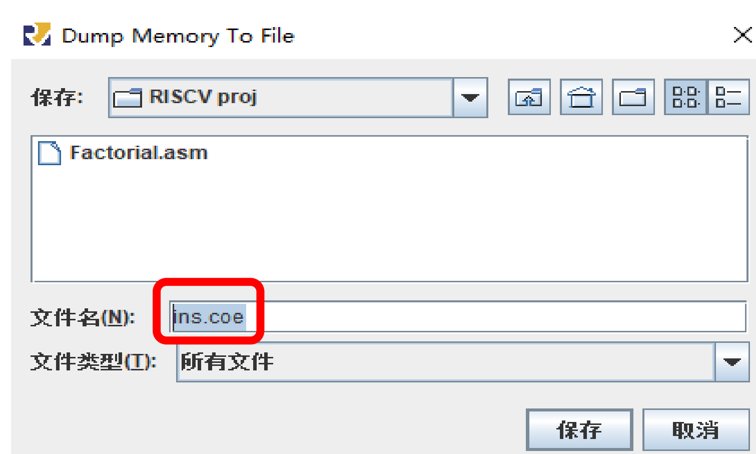
5.RARS:RISC-V Assembler & Runtime Simulator

□ 汇编程序转COE文件

- ✓ 配置存储器：Setting >> Memory Configuration...
- ✓ 汇编程序：Run >> Assemble
- ✓ 导出代码和数据：File >> Dump Memory...







实验原理

5.RARS:RISC-V Assembler & Runtime Simulator

□ 汇编程序转COE文件

✓ **生成COE文件：**

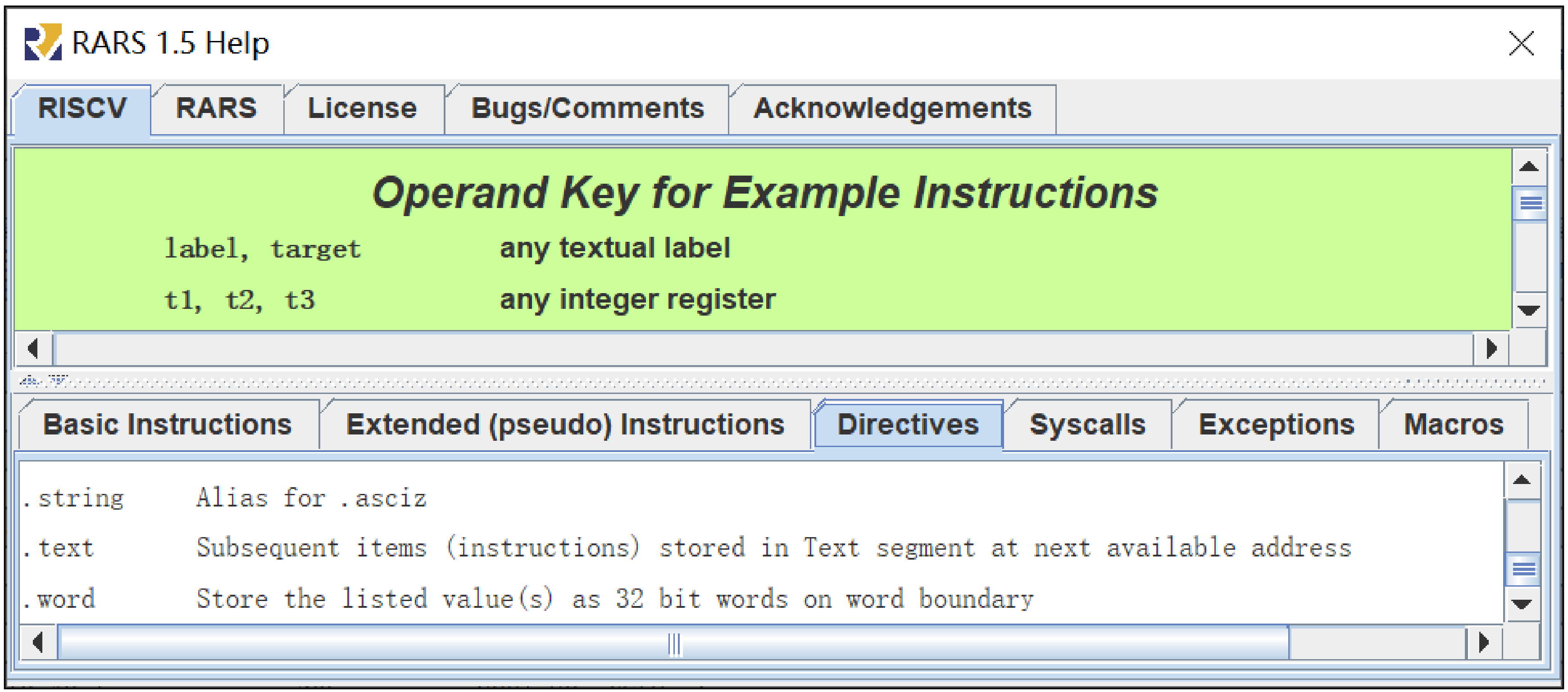
采用记事本分别打开生成的ins.coe和data.coe，在文档的开头添加以下两行语句后保存：

```
memory_initialization_radix  = 16;
memory_initialization_vector =
```

实验原理

5.RARS:RISC-V Assembler & Runtime Simulator

- **help**
  - ✓ RISCV：指令、伪指令、指示符、系统调用……
  - ✓ RARS：IDE、调试、工具……



实验内容

1. 斐波那契数列（6分）

- 编写汇编程序，计算斐波那契数列的第*N*项($1 \le N \le 30$)。初始时，*N*的值保存在 R_2 中。程序执行完成后，数列的第 *N* 项保存在R_3中。

2. 大整数处理（3分）

- 编写汇编程序，计算斐波那契数列的第*N*项($1 \le N \le 80$)。初始时，*N*的值保存在 R_2 中。程序执行完成后，数列的第 *N* 项保存在R_3和R_4中，其中R_3存储结果的高32位，R_4存储结果的低32位。

3. 导出存储器初始化文件（COE文件）（1分）

- 完成上述两项汇编程序后，导出指令段COE文件，以供后续实验使用。

指令架构可选 <u>RV32I</u> 或 <u>LA32R</u>

The End