计算机组成原理

实验三 汇编程序设计

2022春季

zjx@ustc.edu.cn

实验大纲

- **实验目标**
- **实验环境**
- **准备知识**
  - ✓RISC-V汇编指令
  - ✓Ripes软件简介
  - ✓RARS软件简介
- **实验内容**
- **实验步骤**

实验目标

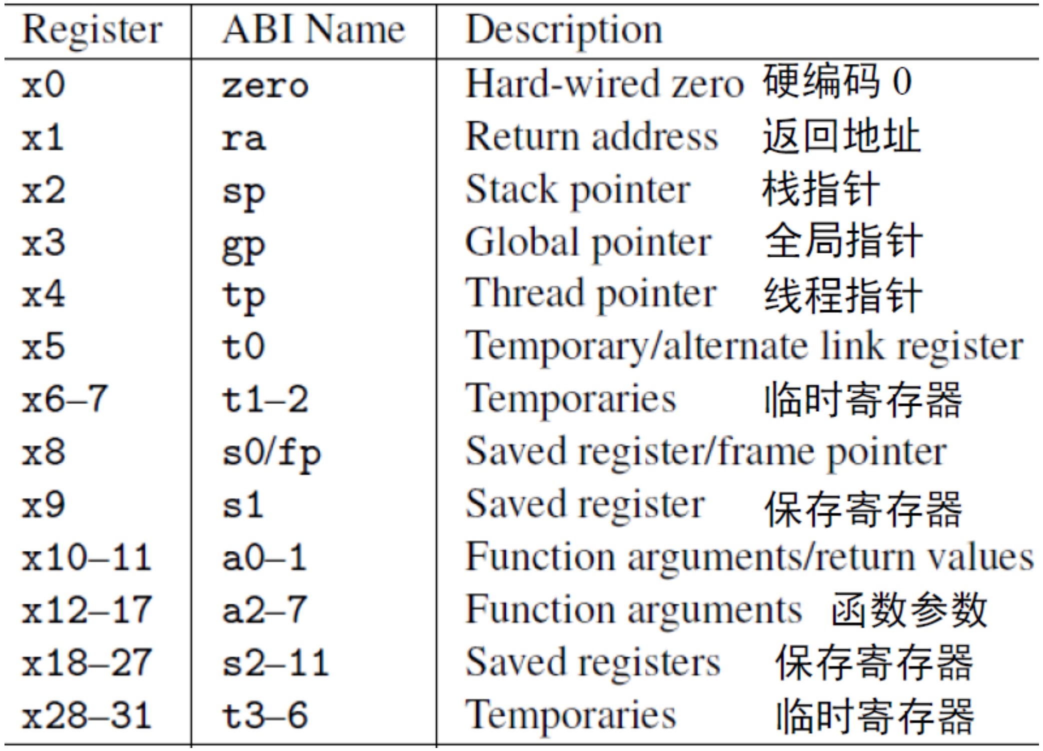
- **熟悉RISC-V汇编指令的格式**
- **熟悉CPU仿真软件Ripes，理解汇编指令执行的基本原理（数据通路和控制器的协调工作过程）**
- **熟悉汇编程序的基本结构，掌握简单汇编程序的设计**
- **掌握汇编仿真软件RARS(RISC-V Assembler & Runtime Simulator)的使用方法，会用该软件进行汇编程序的仿真、调试以及生成CPU测试需要的指令和数据文件（COE）**
- **理解CPU调试模块PDU的使用方法**

实验环境

- PC 一台
- **Ripes：** RISC-V graphical processor simulator
- **Rars：** RISC-V Assembler and Runtime Simulator

准备知识--RISC-V汇编指令

1. RISC-V 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

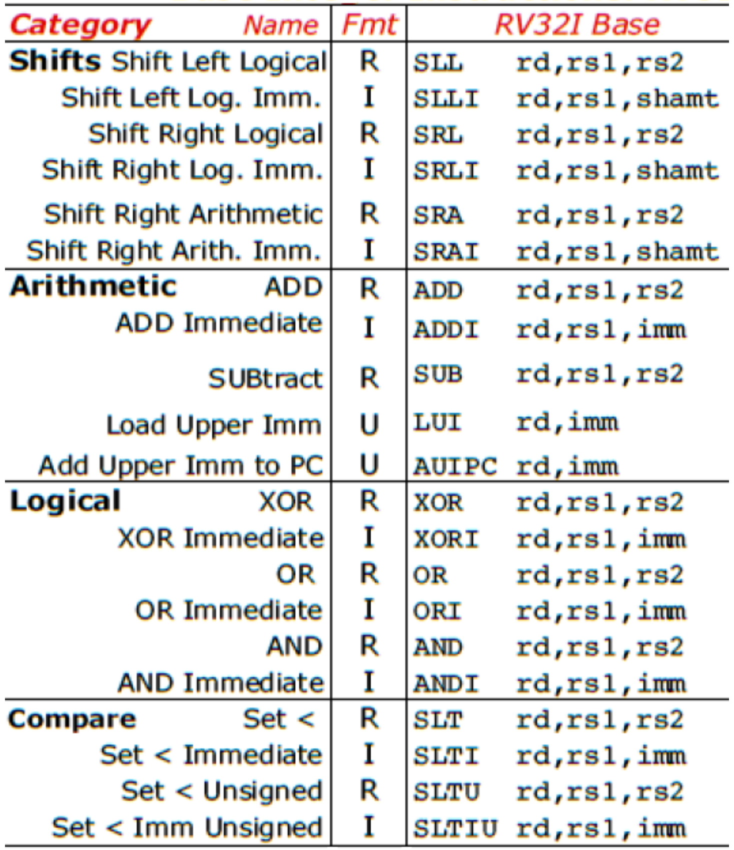
RISC-V整数寄存器的汇编助记符

准备知识--RISC-V汇编指令

2. RV32I指令类型（RISC-V基本32位整数指令集）

1）运算类

- 算术：add, addi, sub, lui, auipc
- 逻辑：and, or, xor
- 移位(shift)：sll, srl, sra
- 比较(set if less than)：slt, sltu



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

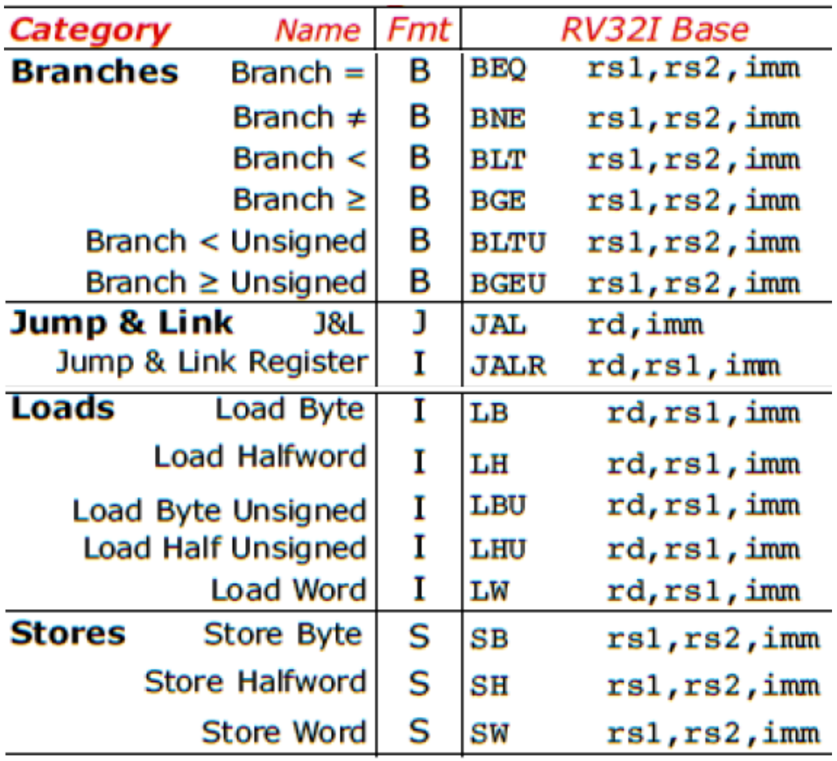
准备知识--RISC-V汇编指令

2)访存类

- 加载(load)：lw, lb, lh, lbu, lhu
- 存储(store)：sw, sb, sh

3)转移类

- 分支(branch)：beq, bne, blt, bge, bltu, 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

准备知识--RISC-V汇编指令

1) 运算指令

➢add rd, rs1, rs2 # x[rd] = x[rs1] + x[rs2]

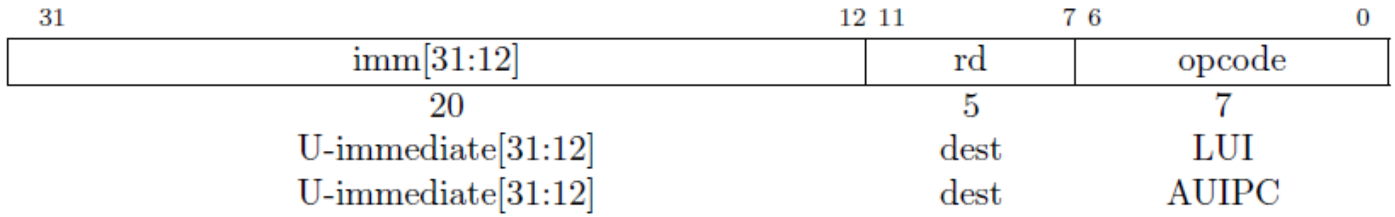
31 25	24 20	19 15	14 12	11 7	6 0
funct7	rs2	rs1	funct3	rd	opcode
7	5	5	3	5	7
0000000	src2	src1	ADD/SLT/SLTU	dest	OP
0000000	src2	src1	AND/OR/XOR	dest	OP
0000000	src2	src1	SLL/SRL	dest	OP
0100000	src2	src1	SUB/SRA	dest	OP

➢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

准备知识--RISC-V汇编指令

- 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

准备知识--RISC-V汇编指令

2) 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

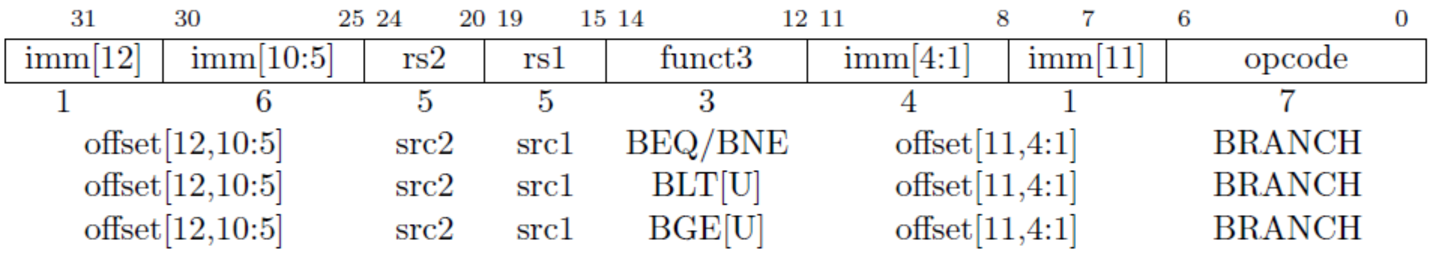
➢sw rs2, offset(rs1)   # M[x[rs1]+sext(offset)=x[rs2]

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

准备知识--RISC-V汇编指令

3) 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

准备知识--RISC-V汇编指令

4) 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

准备知识--RISC-V汇编指令

3. 汇编指示和伪指令

➢**汇编指示符**（Assembler directives）

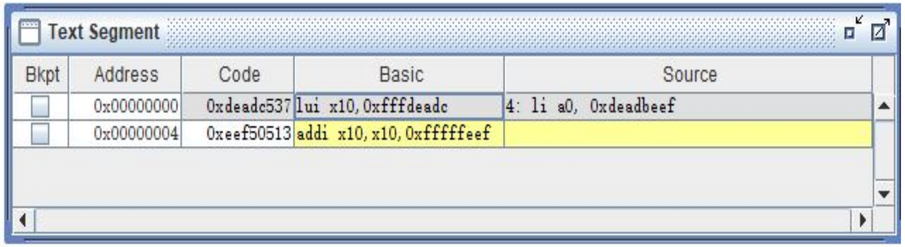
- .data, .text
- .word, .half, .byte, .string
- .eqv
- .align
- ……

Example:
.eqv CONSTANT, 0xdeadbeef
li a0, CONSTANT

# lui a0,0xdeadc
# addi a0,a0,0xffffffeef

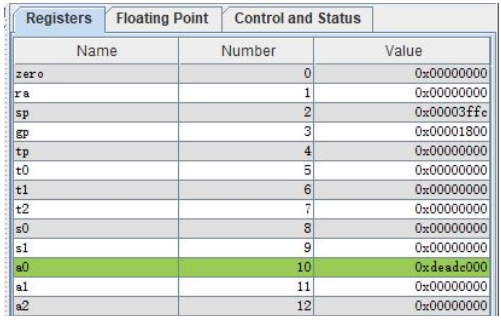
➢ **伪指令**

- li, la, mv
- nop, not, neg
- j, jr, call, ret
- ……



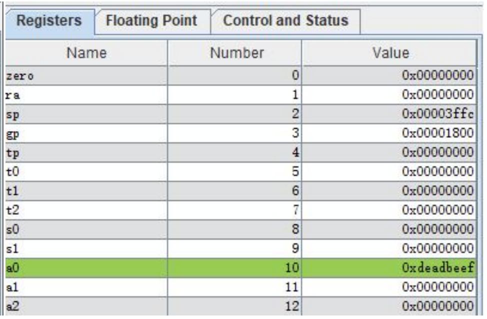
Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00000000	0xdeadc537	lui x10, 0xfffdeadc	4: li a0, 0xdeadbeef
☐	0x00000004	0xeef50513	addi x10, x10, 0xfffffeef	



Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x00003ffc
gp	3	0x00001800
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0xdeadc000
a1	11	0x00000000
a2	12	0x00000000



Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x00003ffc
gp	3	0x00001800
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0xdeadbeef
a1	11	0x00000000
a2	12	0x00000000

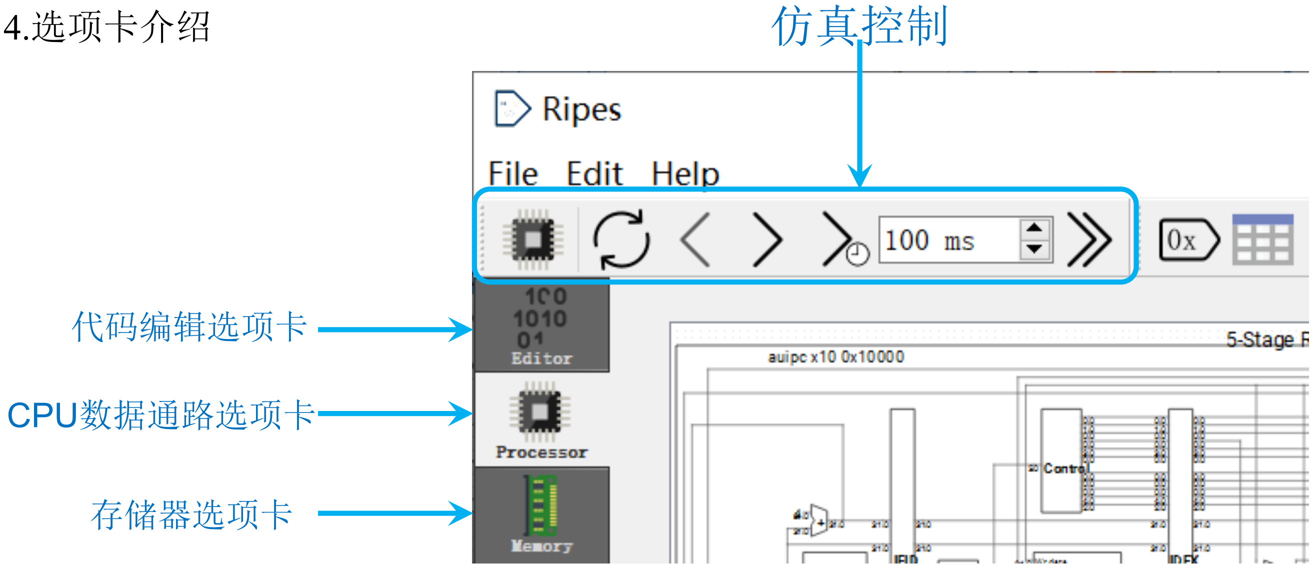
准备知识—Ripes软件简介

1. Ripes is a graphical processor simulator and assembly code editor built for the RISC-V instruction set architecture, suitable for teaching how assembly level code is executed on various microarchitectures.

2. 软件下载链接：https://github.com/mortbopet/Ripes/releases

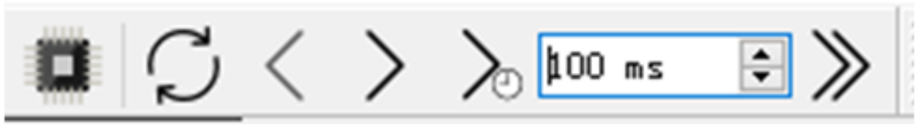
3. 软件简介：https://github.com/mortbopet/Ripes/wiki/Ripes-Introduction

4.选项卡介绍



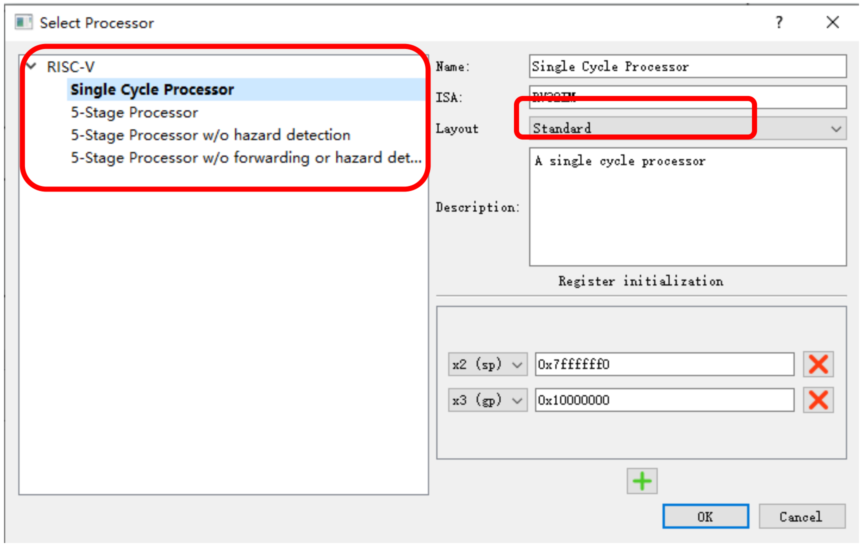
准备知识—Ripes软件简介

➢ 仿真控制



Select Processor　Reset　Reverse　Clock　Auto-clock　Run

✓ Select Processor



Select Processor

- RISC-V
  - **Single Cycle Processor**
  - 5-Stage Processor
  - 5-Stage Processor w/o hazard detection
  - 5-Stage Processor w/o forwarding or hazard det...

Name:	Single Cycle Processor
ISA:	RV32IM
Layout	Standard
Description:	A single cycle processor

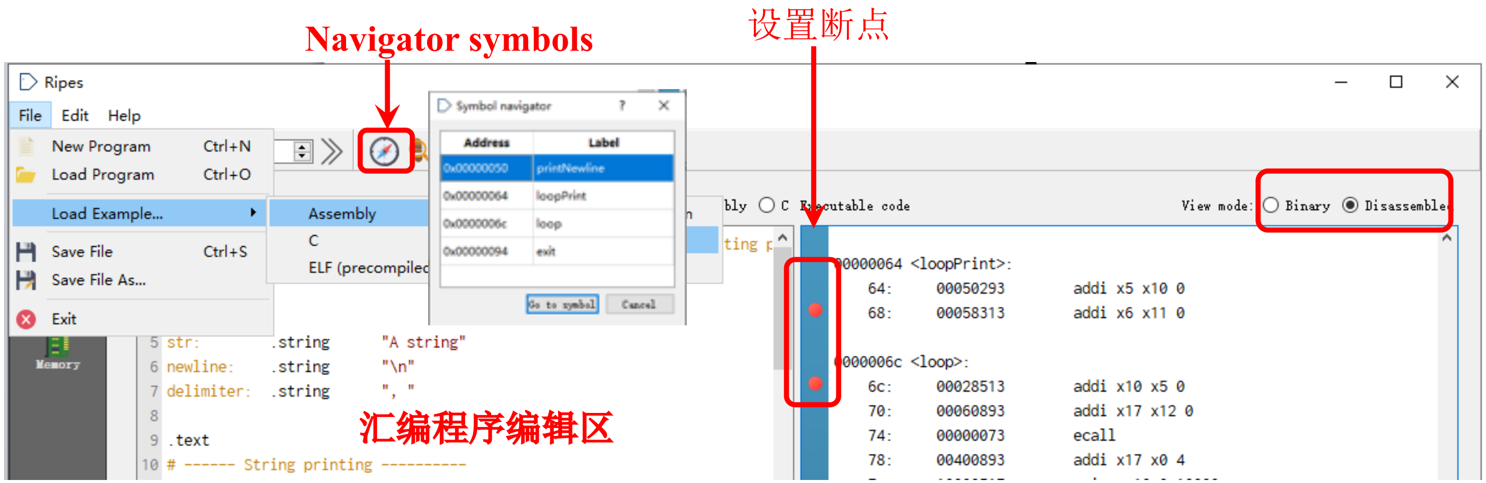
Register initialization

x2 (sp)	0x7ffffff0
x3 (gp)	0x10000000

•**Standard**: Control components and signals are omitted.
•**Extended**: Control components and signals are visible as well as wire bit-widths.

准备知识—Ripes软件简介

➢ 代码编辑选项卡



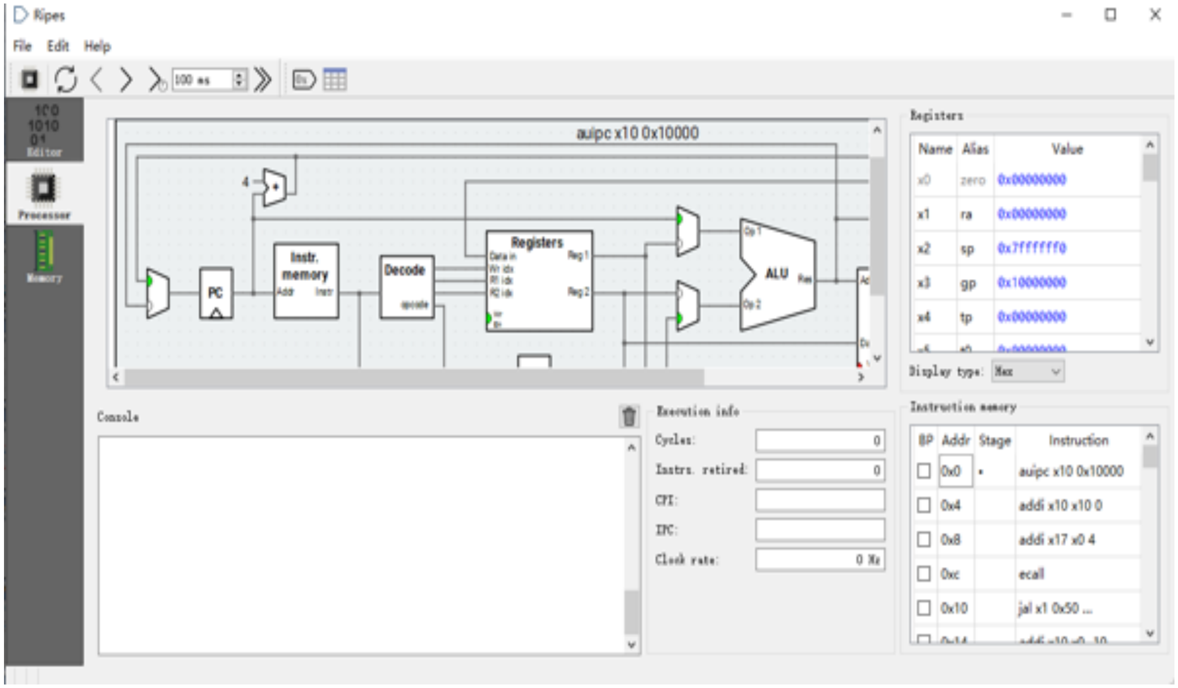
Address	Label
0x00000050	printNewline
0x00000064	loopPrint
0x0000006c	loop
0x00000094	exit

```
str:        .string     "A string"
newline:    .string     "\n"
delimiter:  .string     ", "

.text
# ------ String printing ----------
```

```
0000064 <loopPrint>:
    64:     00050293        addi x5 x10 0
    68:     00058313        addi x6 x11 0

000006c <loop>:
    6c:     00028513        addi x10 x5 0
    70:     00060893        addi x17 x12 0
    74:     00000073        ecall
    78:     00400893        addi x17 x0 4
```

准备知识—Ripes软件简介

➢ CPU数据通路选项卡



Name	Alias	Value
x0	zero	0x00000000
x1	ra	0x00000000
x2	sp	0x7ffffff0
x3	gp	0x10000000
x4	tp	0x00000000

Display type: Hex

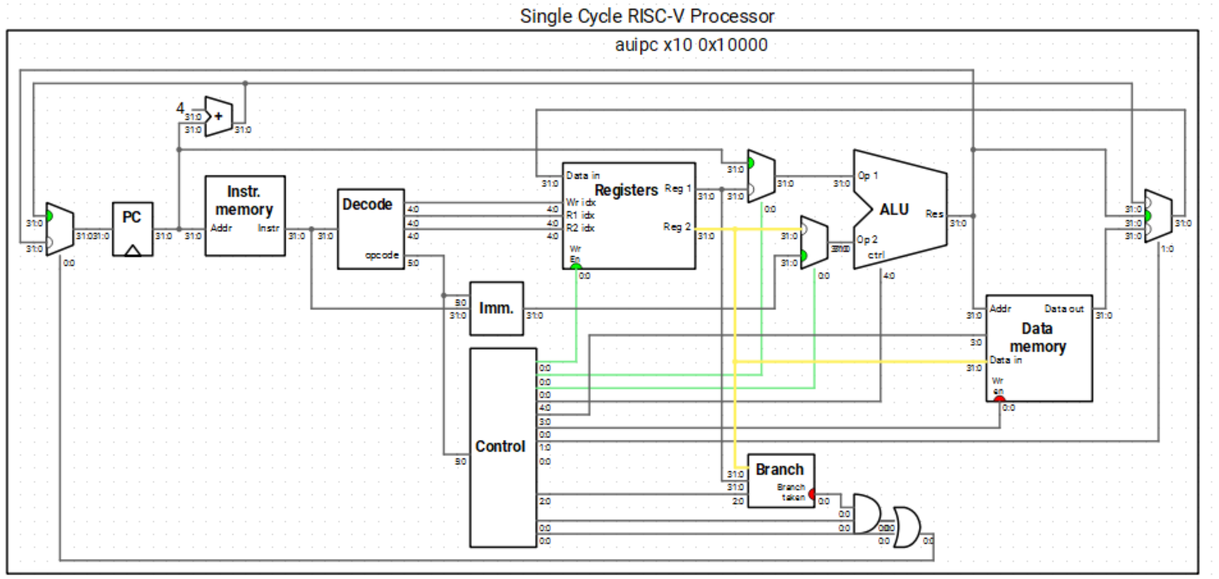
Cycles: 0
Instrs. retired: 0
CPI:
IPC:
Clock rate: 0 Hz

BP	Addr	Stage	Instruction
☐	0x0	•	auipc x10 0x10000
☐	0x4		addi x10 x10 0
☐	0x8		addi x17 x0 4
☐	0xc		ecall
☐	0x10		jal x1 0x50 ...

✓ Registers(can modify)， Instruction memory(BP, PC, stage, instruction)， execution info, console

准备知识—Ripes软件简介

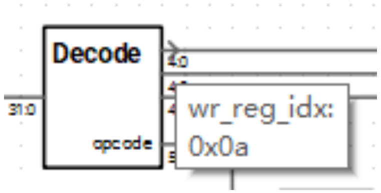
✓ 单周期CPU数据通路



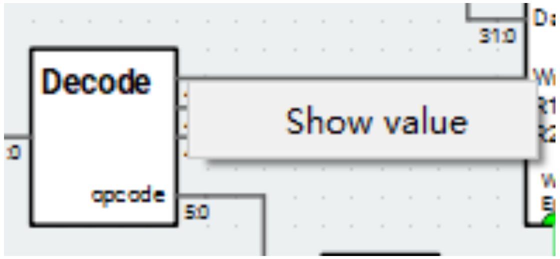
- Green dot in Multiplexers input : signal selected
- Boolean (1-bit signals: high (1) when a wire is green, low (0) when a wire is grey.
- Other signals: when modified, will briefly flash green
- processor view zoom: ctrl+scroll
- Clicking a wire highlights the entirety of the wire: yellow

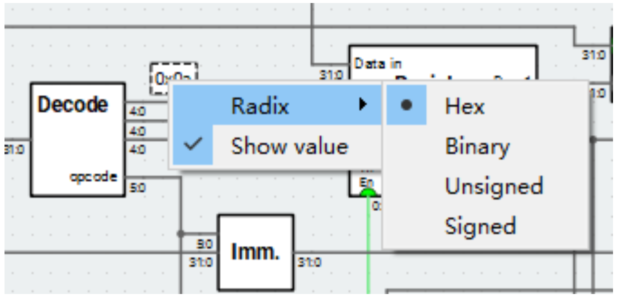
准备知识—Ripes软件简介

- Hover over any port in the processor view: display the name and value of the port



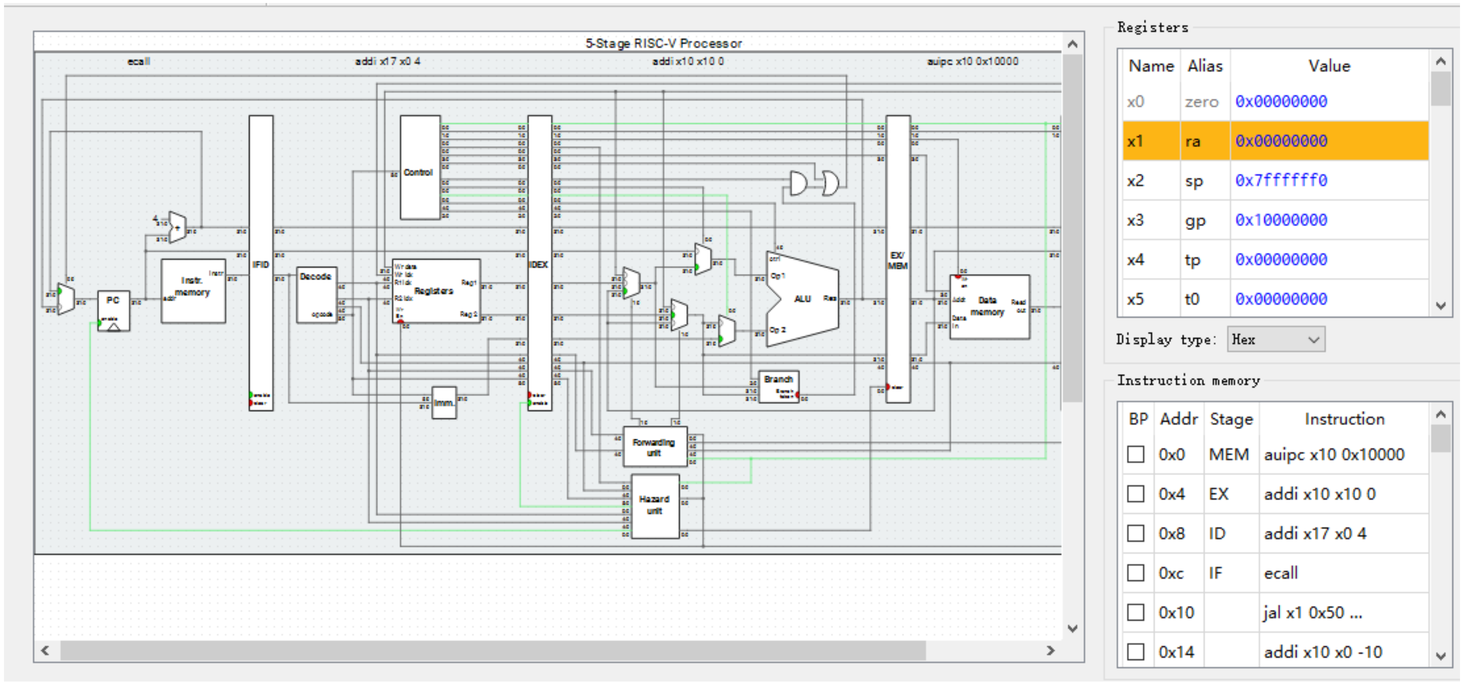
- right click on any port and press "*show value*" to display its label.
- If a port's value label has been made visible, it is possible to change the radix of the displayed value through right-clicking the port label.





准备知识—Ripes软件简介

✓ 流水线CPU数据通路



Registers

Name	Alias	Value
x0	zero	0x00000000
x1	ra	0x00000000
x2	sp	0x7ffffff0
x3	gp	0x10000000
x4	tp	0x00000000
x5	t0	0x00000000

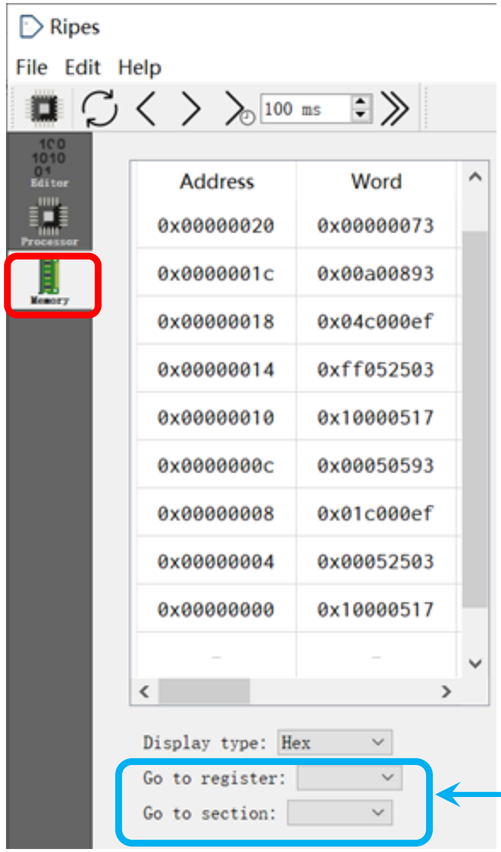
Display type: Hex

Instruction memory

BP	Addr	Stage	Instruction
☐	0x0	MEM	auipc x10 0x10000
☐	0x4	EX	addi x10 x10 0
☐	0x8	ID	addi x17 x0 4
☐	0xc	IF	ecall
☐	0x10		jal x1 0x50 ...
☐	0x14		addi x10 x0 -10

准备知识—Ripes软件简介

➢ 存储器选项卡



Address	Word
0x00000020	0x00000073
0x0000001c	0x00a00893
0x00000018	0x04c000ef
0x00000014	0xff052503
0x00000010	0x10000517
0x0000000c	0x00050593
0x00000008	0x01c000ef
0x00000004	0x00052503
0x00000000	0x10000517

Display type: Hex
Go to register:
Go to section:

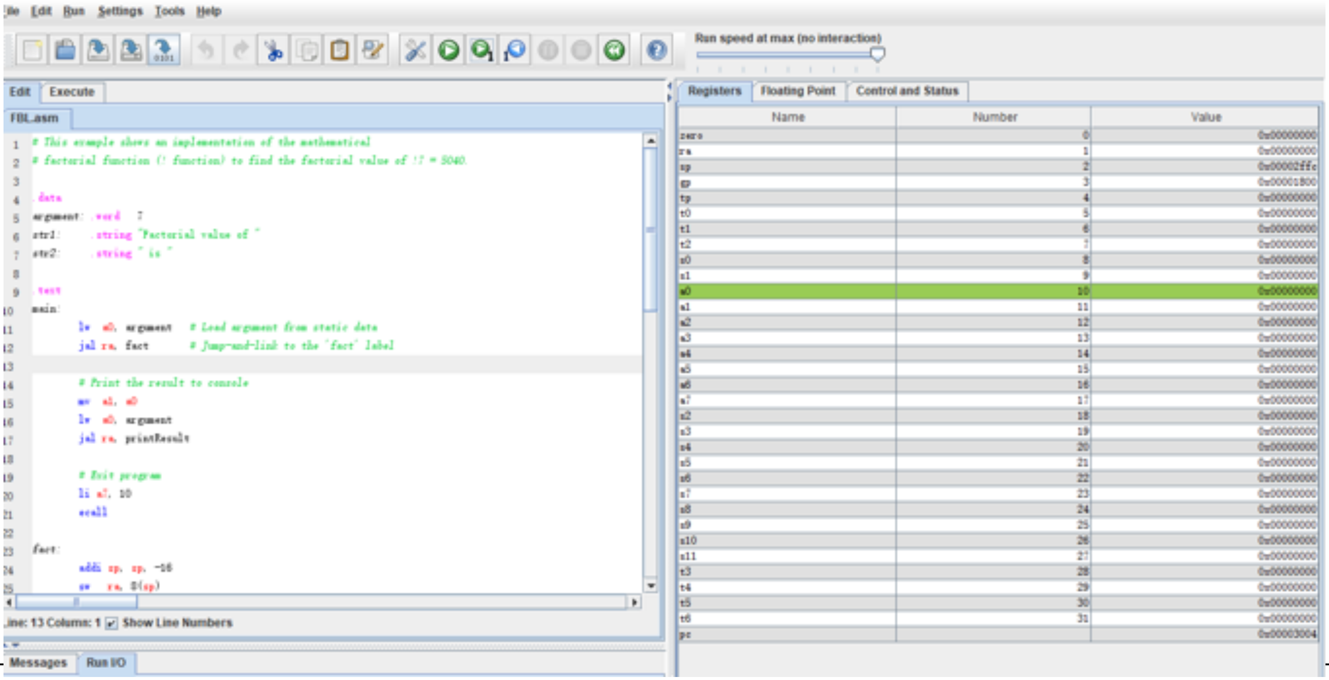
存储器定位

准备知识—Rars软件简介

1. RARS is written in Java and requires at least Release 1.8 of the Java SE Java Runtime Environment (JRE) to work. It is distributed as an executable JAR file.

2. 软件简介： https://github.com/TheThirdOne/rars/issues

3.界面介绍



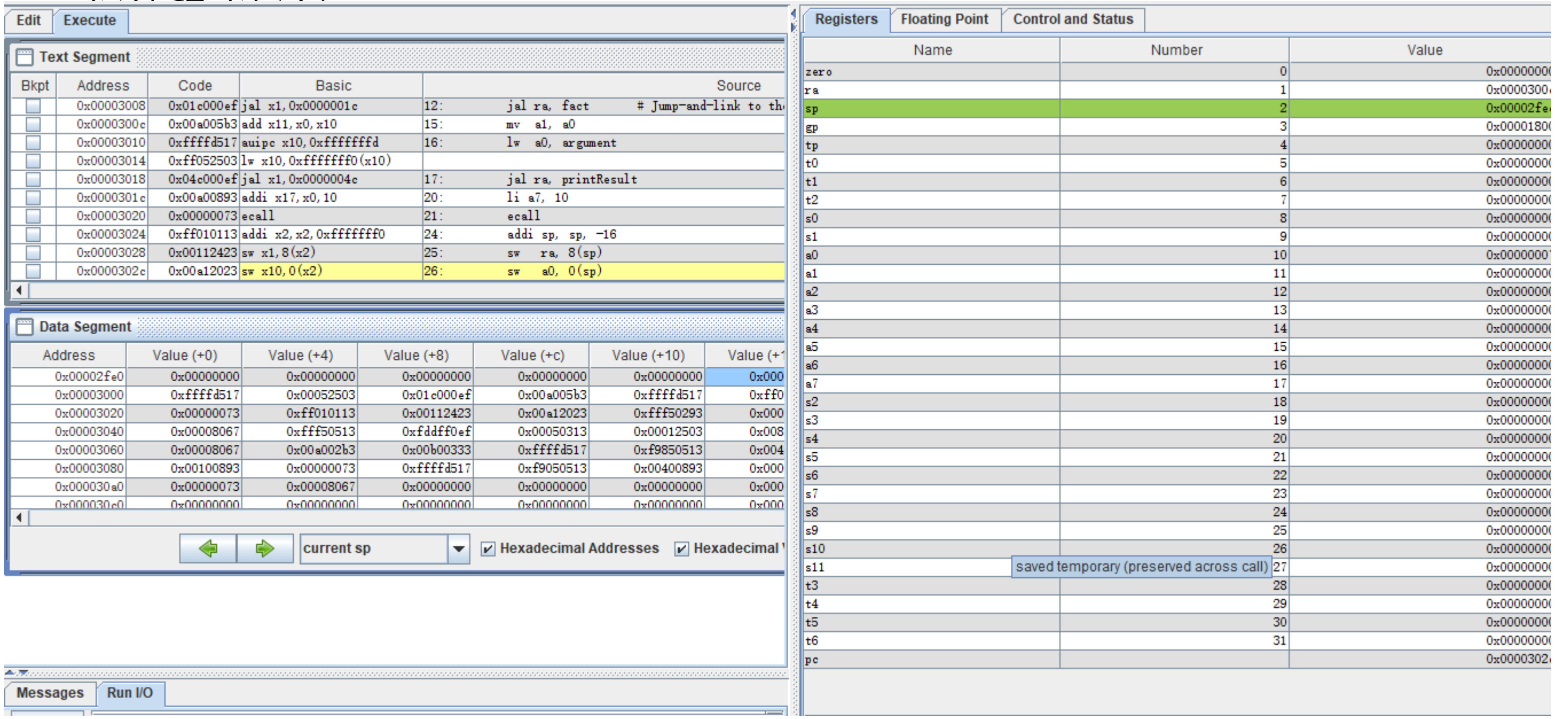
```
# This example shows an implementation of the mathematical
# factorial function (! function) to find the factorial value of !7 = 5040.

.data
argument: .word   7
str1:       .string "Factorial value of "
str2:       .string " is "

.text
main:
          lw  a0, argument   # Load argument from static data
          jal ra, fact       # Jump-and-link to the 'fact' label

          # Print the result to console
          mv  a1, a0
          lw  a0, argument
          jal ra, printResult

          # Exit program
          li a7, 10
          ecall

fact:
          addi sp, sp, -16
          sw   ra, 8(sp)
```

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x00002ffc
gp	3	0x00001800
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00003004

准备知识—Rars软件简介

4.存储器设置

Settings>>Memory Configuration>>compact, data at Address0>>Apply and Close



Bkpt	Address	Code	Basic	Source
☐	0x00003008	0x01c000ef	jal x1, 0x0000001c	12: jal ra, fact # Jump-and-link to th
☐	0x0000300c	0x00a005b3	add x11, x0, x10	15: mv a1, a0
☐	0x00003010	0xffffd517	auipc x10, 0xfffffffd	16: lw a0, argument
☐	0x00003014	0xff052503	lw x10, 0xfffffff0(x10)	
☐	0x00003018	0x04c000ef	jal x1, 0x0000004c	17: jal ra, printResult
☐	0x0000301c	0x00a00893	addi x17, x0, 10	20: li a7, 10
☐	0x00003020	0x00000073	ecall	21: ecall
☐	0x00003024	0xff010113	addi x2, x2, 0xfffffff0	24: addi sp, sp, -16
☐	0x00003028	0x00112423	sw x1, 8(x2)	25: sw ra, 8(sp)
☐	0x0000302c	0x00a12023	sw x10, 0(x2)	26: sw a0, 0(sp)

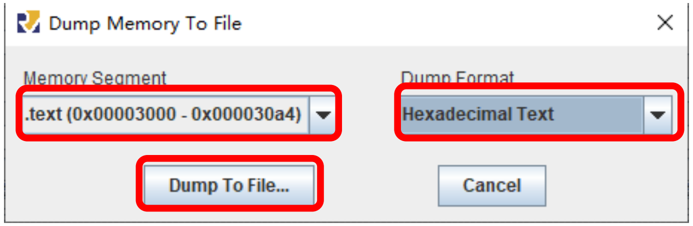
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+1
0x00002fe0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x000
0x00003000	0xffffd517	0x00052503	0x01c000ef	0x00a005b3	0xffffd517	0xff0
0x00003020	0x00000073	0xff010113	0x00112423	0x00a12023	0xfff50293	0x000
0x00003040	0x00008067	0xfff50513	0xfddff0ef	0x00050313	0x00012503	0x008
0x00003060	0x00008067	0x00a002b3	0x00b00333	0xffffd517	0xf9850513	0x004
0x00003080	0x00100893	0x00000073	0xffffd517	0xf9050513	0x00400893	0x000
0x000030a0	0x00000073	0x00008067	0x00000000	0x00000000	0x00000000	0x000
0x000030c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x000

Name	Number	Value
zero	0	0x0000000
ra	1	0x0000300
sp	2	0x00002fe
gp	3	0x0000180
tp	4	0x0000000
t0	5	0x0000000
t1	6	0x0000000
t2	7	0x0000000
s0	8	0x0000000
s1	9	0x0000000
a0	10	0x0000000
a1	11	0x0000000
a2	12	0x0000000
a3	13	0x0000000
a4	14	0x0000000
a5	15	0x0000000
a6	16	0x0000000
a7	17	0x0000000
s2	18	0x0000000
s3	19	0x0000000
s4	20	0x0000000
s5	21	0x0000000
s6	22	0x0000000
s7	23	0x0000000
s8	24	0x0000000
s9	25	0x0000000
s10	26	0x0000000
s11	saved temporary (preserved across call) 27	0x0000000
t3	28	0x0000000
t4	29	0x0000000
t5	30	0x0000000
t6	31	0x0000000
pc		0x0000302

准备知识—Rars软件简介

6.代码段段机器码(16进制)导出

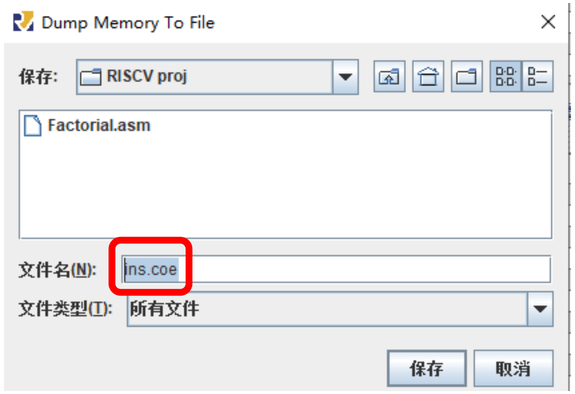
File>>Dump Memory To File，按下图设置完毕后，选择文件保存路径，命名为ins.coe。



Dump Memory To File

Memory Segment: .text (0x00003000 - 0x000030a4)

Dump Format: Hexadecimal Text



Dump Memory To File

保存: RISCV proj

Factorial.asm

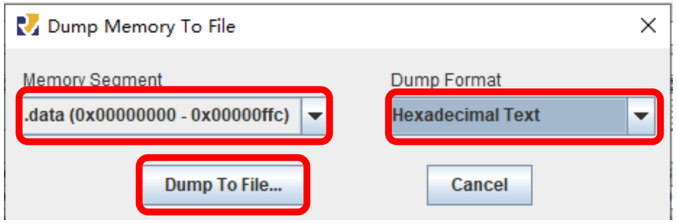
文件名(N): ins.coe

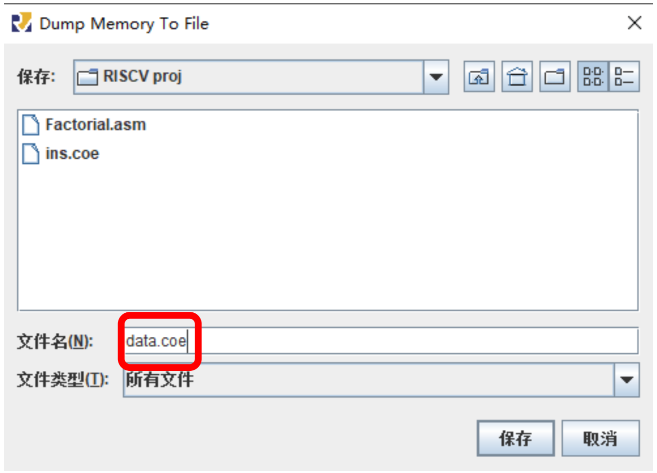
文件类型(T): 所有文件

准备知识—Rars软件简介

7.数据段机器码(16进制)导出

File>>Dump Memory To File，按下图设置完毕后，选择文件保存路径，命名为data.coe。


Dump Memory To File
Memory Segment: .data (0x00000000 - 0x00000ffc)
Dump Format: Hexadecimal Text


Dump Memory To File
保存: RISCV proj
Factorial.asm
ins.coe
文件名(N): data.coe
文件类型(T): 所有文件

8. 采用记事本分别打开生成的ins.coe和data.coe，在文档的最开始加上以下语句后保存：memory_initialization_radix = 16;

memory_initialization_vector =

实验内容

1.理解并仿真RIPES示例汇编程序

加载Ripes示例汇编程序 (Console Printing)→选择单周期CPU数据通路→单步执行程序→观察数据通路控制信号和寄存器内容的变化

2.设计汇编程序，验证6条指令功能

Rars软件设计汇编程序→单步运行程序→人工检查→生成COE文件

✓sw, lw

✓add, addi

✓beq, jal

备注：通过查看数据存储器和32个通用寄存器来实现人工检查

实验内容

示例：

```
.data
out: .word 0xff      #led, 初始全亮
in: .word 0                   #switch

.text
la a0, out                    #仿真需要
sw x0, 0(a0)                  #test sw: 全灭led
addi t0, x0, 0xff    #test addi: 全亮led
sw t0, 0(a0)
lw t0, 4(a0)                  #test lw: 由switch设置led
sw t0, 0(a0)
… …
```

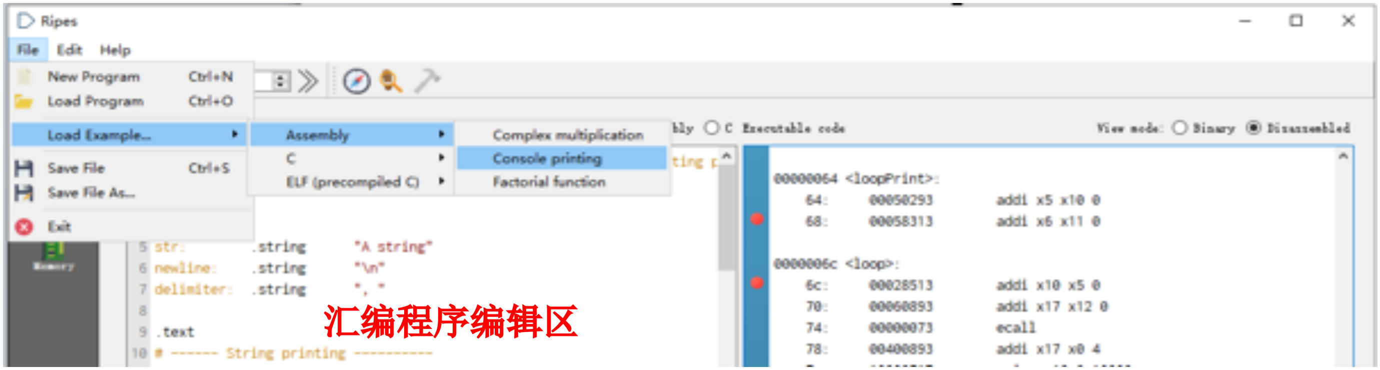
3. 设计汇编程序，计算斐波那契—卢卡斯数列

➢数列前两项：1，2

➢数据输出方式：数据存储器

实验步骤

1. 理解并仿真RIPES示例汇编程序 (Console Printing)



```
str:        .string     "A string"
newline:    .string     "\n"
delimiter:  .string     ", "

.text
# ------ String printing ----------
```

Executable code   View mode: ○ Binary [✓ Disassembled]

```
00000064 <loopPrint>:
    64:    00050293        addi x5 x10 0
    68:    00058313        addi x6 x11 0

0000006c <loop>:
    6c:    00028513        addi x10 x5 0
    70:    00060893        addi x17 x12 0
    74:    00000073        ecall
    78:    00400893        addi x17 x0 4
```

2. Rars软件设计汇编程序，实现人工检查6条指令功能，并生成COE文件

- ✓ sw, lw
- ✓ add, addi
- ✓ beq, jal

3. Rars软件设计汇编程序，实现计算斐波那契—卢卡斯数列（数列前两项为1，2），并生成COE文件

The End