

实验报告

实验名称：RISC-V 基本整数指令集汇编器设计与实现

班级：智能 1602

学号：201608010609

姓名：李鹏飞

实验目标

设计一个 RISC-V 基本整数指令集汇编器，能够实现汇编指令向二进制的转化。

实验要求

- 采用 C/C++ 编写程序
- 汇编器的输入是模拟的汇编指令文件
- 汇编器的输出是汇编指令经过汇编之后的二进制指令文件

实验内容

1. 汇编器简介

汇编器（Assembler）是将汇编语言翻译为机器语言的程序。一般而言，汇编生成的是目标代码，需要经链接器（Linker）生成可执行代码才可以执行。

汇编语言是一种以处理器指令系统为基础的低级语言，采用助记符表达指令操作码，采用标识符表示指令操作数。作为一门语言，对应于高级语言的编译器，需要一个“汇编器”来把汇编语言原文件汇编成机器可执行的代码。

2.RISC-V 指令集内容

我们在这里编写的是 RV32I 指令集，其包含了六种基本指令格式，分别是：

用于寄存器-寄存器操作的 R 类型指令，用于短立即数和访存 load 操作的 I 型指令，用于访存 store 操作的 S 型指令，用于条件跳转操作的 B 类型指令，用于长立即数的 U 型指令和用于无条件跳转的 J 型指令。

3.RISC-V 指令集编码格式

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	

4.RISC-V 指令

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
Category	Name	Fmt	RV32I Base
Compare			
Set<	R	SLT	rd,rs1,rs2
Set<Immediate	I	SLTI	rd,rs1,rs2
Set<Unsigned	R	SLTU	rd,rs1,rs2
Set<Imm Unsigned	I	SLTIU	rd,rs1,imm
Branches			
Branch=	B	BEQ	rs1,rs2,imm
Branch \neq	B	BNE	rs1,rs2,imm
Branch<	B	BLT	rs1,rs2,imm
Branch \geq	B	BGE	rs1,rs2,imm
Branch<Unsigned	B	BLTU	rs1,rs2,imm
Branch \geq Unsigned	B	BGEU	rs1,rs2,imm
Jump&Link			
J&L	J	JAL	rd,imm
Jump&Link Register	I	JALR	rd,rs1,imm
Synch			
Synch thread	I	FENCE	
Synch Instr&Data	I	FENCE.I	
Environment			
CALL	I	ECALL	
BREAK	I	EBREAK	
Control Status Register(CSR)			
Read/Write	I	CSRRW	rd,csr,rs1
Read&Set Bit	I	CSRRS	rd,csr,rs1
Read&Clear Bit	I	CSRRC	rd,csr,rs1
Read/Write Imm	I	CSRRWI	rd,csr,imm
Read&Set Bit Imm	I	CSRRSI	rd,csr,imm
Read&Clear Bit Imm	I	CSRRCI	rd,csr,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

汇编器的执行流程

步骤 1：编程者用文本编辑器创建一个 ASCII 文本文件，称之为源文件。

步骤 2：汇编器读取源文件，并生成目标文件，即对程序的机器语言翻译。或者，它也会生成列表文件。只要出现任何错误，编程者就必须返回步骤 1，修改程序。

步骤 3：链接器读取并检查目标文件，以便发现该程序是否包含了任何对链接库中过程的调用。链接器从链接库中复制任何被请求的过程，将它们与目标文件组合，以生成可执行文件。

步骤 4：操作系统加载程序将可执行文件读入内存，并使 CPU 分支到该程序起始地址，然后程序开始执行。

汇编器程序框架

我们将模拟器的框架设计如下：

```
<标号>:add x1, x2, x3
```

```
<标号>:10101010...
```

汇编程序文件 file.asm

一行一个汇编语句

```
初始化地址计数器 addr_counter = 0;
```

```
while(file.asm 没有到文件尾){
```

```
    读入一行
```

```
    while(读入的是纯标号且不是文件尾){ 继续读一行 }
```

```
    拆开行，得到标号（有可能没有），操作码或者伪指令助记符，操作数
```

```
    if(有标号){ 记下标号和当前地址计数器的值，保存到符号表；
```

```
        查看未决汇编语句是否需要这个标号，并解决
```

```
    }
```

```
    if(操作码助记符){
```

```
        生成操作码编码;
```

操作数 -> 寄存器编号或者立即数

```
if(操作数是标号){ 查找符号表，如果查到，计算得到偏移量；  
    如果没查到，记下当前汇编语句和地址  
}
```

生成指令的二进制表示 }

```
else(伪指令助记符){ 根据伪指令含义执行相应转换 }  
}
```

测试

测试平台

模拟器在如下机器上进行了测试

部件	配置	备注
CPU	Core i5-6700U	
内存	DDR4 12GB	
操作系统	Windows10 家庭版	

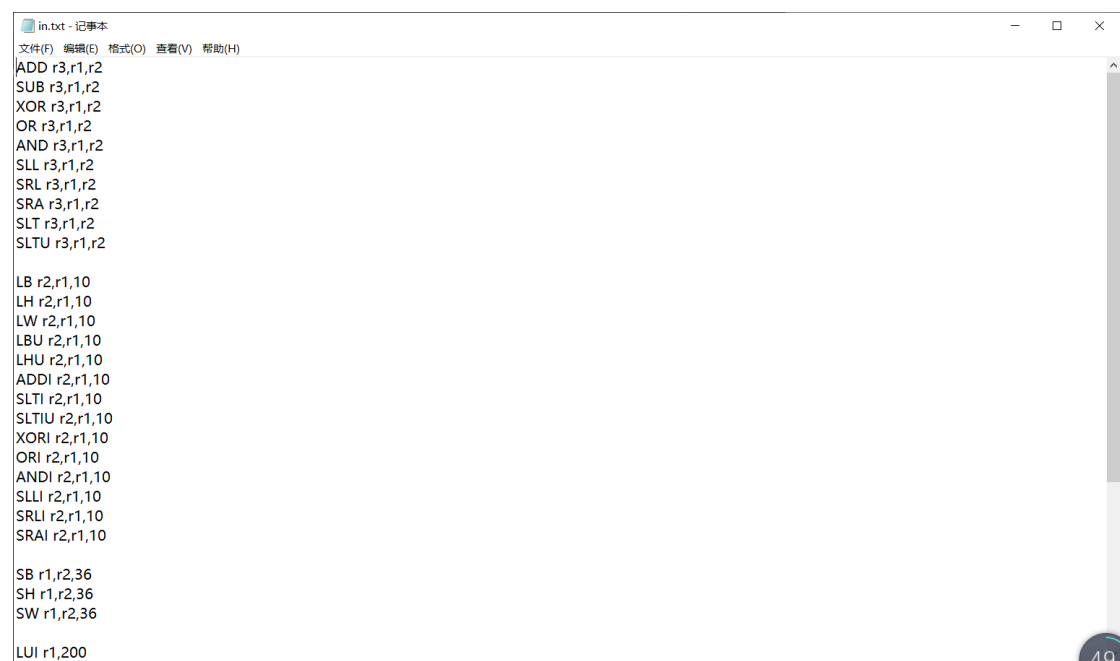
测试记录

模拟器输入如下：

```
ADD r3,r1,r2  
SUB r3,r1,r2  
XOR r3,r1,r2  
OR r3,r1,r2  
AND r3,r1,r2  
SLL r3,r1,r2  
SRL r3,r1,r2  
SRA r3,r1,r2  
SLT r3,r1,r2  
SLTU r3,r1,r2  
LB r2,r1,10  
LH r2,r1,10  
LW r2,r1,10  
LBU r2,r1,10  
LHU r2,r1,10  
ADDI r2,r1,10  
SLTI r2,r1,10
```

```
SLTIU r2,r1,10
XORI r2,r1,10
ORI r2,r1,10
ANDI r2,r1,10
SLLI r2,r1,10
SRLI r2,r1,10
SRAI r2,r1,10
SB r1,r2,36
SH r1,r2,36
SW r1,r2,36
LUI r1,200
AUIPC r1,200
BEQ r1,r2,200
BNE r1,r2,200
BLT r1,r2,200
BGE r1,r2,200
BLTU r1,r2,200
BGEU r1,r2,200
JAL r1,100
JALR r2,r1,100
```

我们将输入放入 in.txt 文件中，利用 freopen 函数进行读入



```
in.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
ADD r3,r1,r2
SUB r3,r1,r2
XOR r3,r1,r2
OR r3,r1,r2
AND r3,r1,r2
SLL r3,r1,r2
SRL r3,r1,r2
SRA r3,r1,r2
SLT r3,r1,r2
SLTU r3,r1,r2

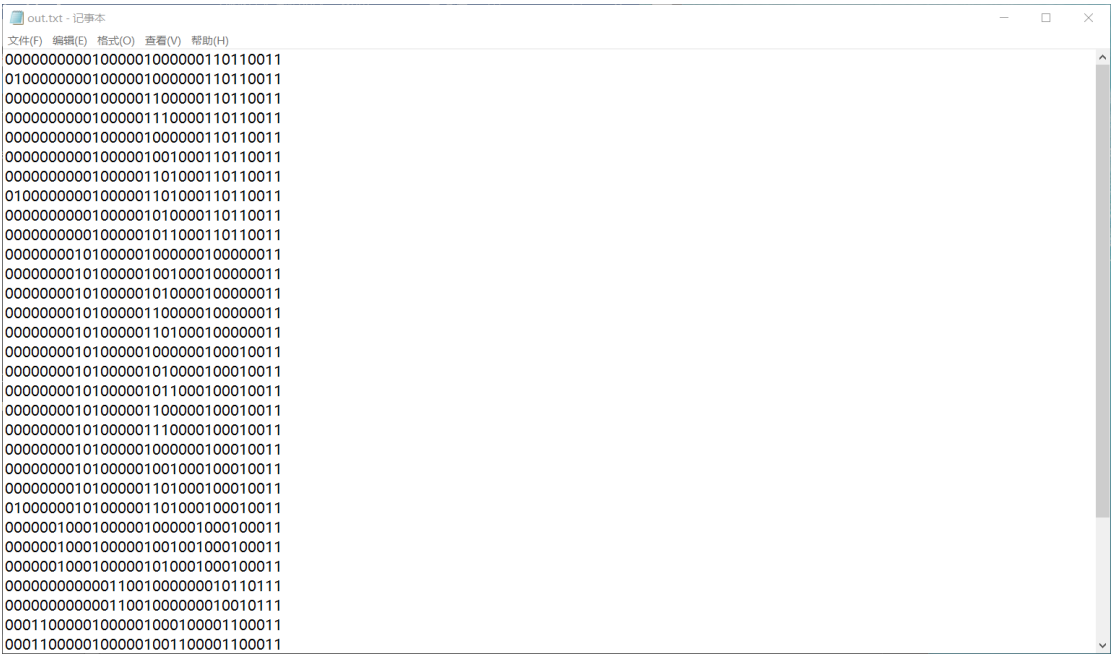
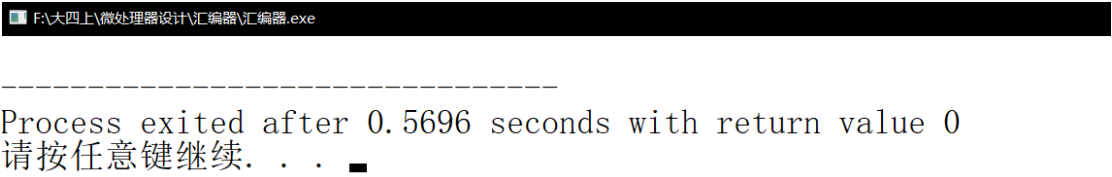
LB r2,r1,10
LH r2,r1,10
LW r2,r1,10
LBU r2,r1,10
LHU r2,r1,10
ADDI r2,r1,10
SLTI r2,r1,10
SLTIU r2,r1,10
XORI r2,r1,10
ORI r2,r1,10
ANDI r2,r1,10
SLLI r2,r1,10
SRLI r2,r1,10
SRAI r2,r1,10

SB r1,r2,36
SH r1,r2,36
SW r1,r2,36

LUI r1,200
```

共 37 条指令。

汇编器运行的截图如下，我们将结果输入至 out.txt 文件中：



将 37 条指令进行汇编操作，全部输出。

分析和结论

从测试记录来看，汇编器实现了对输入汇编指令的读入、汇编、输出操作，基本实现了最初的实验要求。

根据分析结果，可以认为编写的汇编器实现了要求的功能，完成了实验目标。

实验心得体会

因为我们专业没有学习编译原理，所以对编译器的流程和原理一开始还是比较陌生的，通过这个实验有了一些了解。在今后也会努力做一些课外的拓展，学习新知识。