汇编器实验报告

班级：物联 1601

学号：201608010628

姓名：曾彤芳

实验任务

完成一个模拟 RISC-V 的基本整数指令集 RV32I 的汇编器设计

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

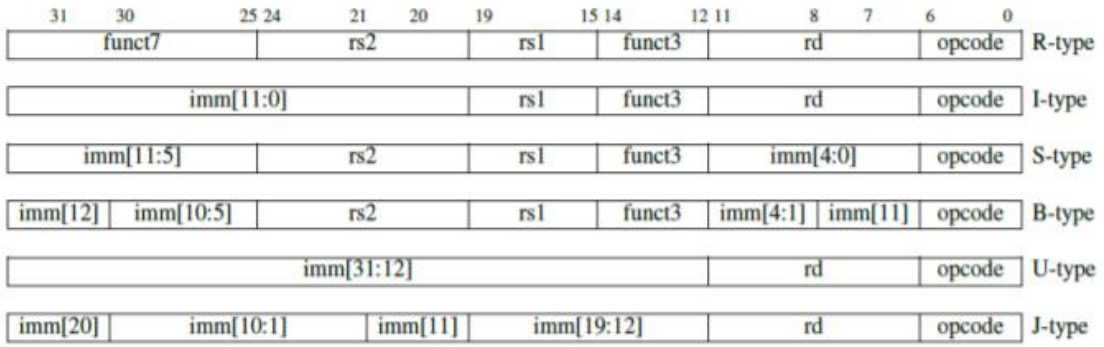
实验内容

汇编器介绍

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。常用的高级语言编译器有 Microsoft 公司的 MASM 系列和 Borland 公司的 TASM 系列编译器，还有一些小公司推出的或者免费的汇编软件包等。

RISC-V 指令集编码格式



RISC-V 指令

基本指令格式

四种基础指令格式 R/I/S/U

imm：立即数

rs1：源寄存器 1

rs2：源寄存器 2

rd：目标寄存器

opcode：操作码

整数计算

ADDI：将 12 位有符号立即数和 rs 相加，溢出忽略，直接使用结果的最低 32bit，并存入 rd

SLTI：如果 rs 小于立即数(都是有符号整数),将 rd 置 1,否则置 0

SLTIU：和 SLTI 一致，不过都是无符号数

ANDI/ORI/XORI：rs 与有符号 12 位立即数进行 and,or,xor 操作

ADD/SUB:rs1(+/-)rs2 => rd

SLT/SLTU: 如果 rs1<rs2，rd 写 1; 否则 rd 为 0

AND/OR/XOR: rs1 与 rs2 进行 and,or,xor 操作

SLL/SRL/SRA: 和"寄存器-立即数"指令一致，将 r2 的低 5 位作为立即数即可

NOP 指令:

实际上是 ADDI x0,x0,0

JAL：J 类指令，立即数+pc 为跳转目标，rd 存放 pc+4（返回地址）

跳转范围为 pc(+/-)1MB

JALR：I 类指令，rs+立即数为跳转目标，rd 存放 pc+4（返回地址）

实现远跳转

BEQ/BNE：rs1(==/!=)rs2，分别在相等或者不等时，发生跳转

BLT：rs1 < rs2，跳转

BGE：rs1 >= rs2，跳转

LOAD：rs 作为基地址，加上有符号的偏移，读取到 rd 寄存器

STORE：rs1 作为基地址加上有符号的偏移，作为内存地址，写入内容为 rs2

CSRRW：Atomic Read/Write CSR

读取 CSR 的值存入 rd 寄存器，并将 rs 存入 CSR

CSRRS：Atomic Read and Set Bits in CSR

读取 CSR 的值存入 rd 寄存器，并根据 rs 中高位对 CSR 置 1

CSRRC：Atomic Read and Clear Bits in CSR

读取 CSR 的值存入 rd 寄存器，并根据 rs 中高位对 CSR 置 0

CSRRWI/CSRRSI/CSRRCI

将 CSRRW 类寄存器中的 rs 换成立即数

RDCYCLE：时钟周期计数

RDTIME：时间 tick 数

RDINSTRET：指令数

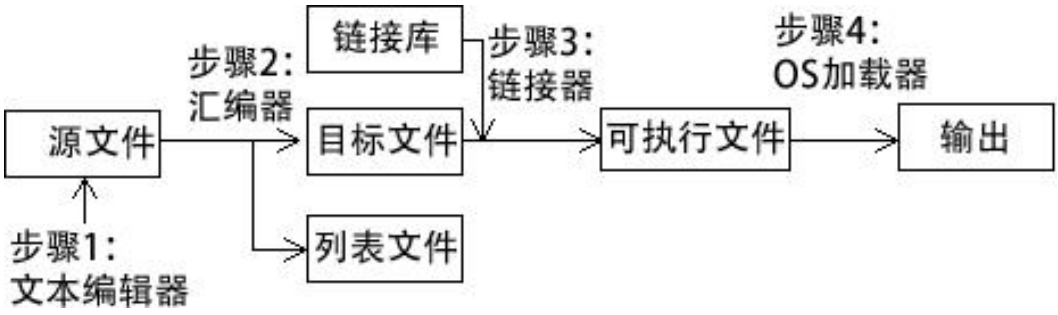
ECALL

EBREAK

汇编器设计

汇编器实现的关键三点：汇编指令的表示、二进制指令的表示、汇编指令到二进制指令之间的转换三个方面。

在程序中我们对输入的汇编指令是按照空白符间隔的方式进行的汇编代码切分，如果进一步改进，可以对汇编代码进行词法分析，切分出汇编 token——指令码和操作数，然后将指令码和操作数翻译成对应的二进制代码。



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。

<标号>: add x1, x2, x3
<标号>:
10101010101101001

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

模块	配置	备注
CPU	Core i5-6700U	
操作系统	Windows10	

测试结果

输入：

```
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,r1,8

LH r2,r1,8

LW r2,r1,8

LBU r2,r1,8

LHU r2,r1,8

ADDI r2,r1,8

SLTI r2,r1,8

SLTIU r2,r1,8

XORI r2,r1,8

ORI r2,r1,8

ANDI r2,r1,8

SLLI r2,r1,8

SRLI r2,r1,8

SRAI r2,r1,8

SB r1,r2,24

SH r1,r2,24

SW r1,r2,24

LUI r1,30

AUIPC r1,30

BEQ r1,r2,30

BNE r1,r2,30

BLT r1,r2,30

BGE r1,r2,30

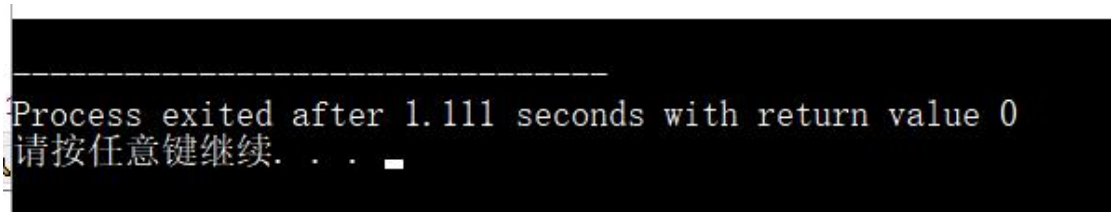
BLTU r1,r2,30

BGEU r1,r2,30

JAL r1,15

JALR r2,r1,15

运行：



结果：

output.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
00000000001000001000000110110011
01000000001000001000000110110011
00000000001000001100000110110011
00000000001000001110000110110011
00000000001000001000000110110011
00000000001000001001000110110011
00000000001000001101000110110011
01000000001000001101000110110011
00000000001000001010000110110011
00000000001000001011000110110011
00000000100000001000000100000011
00000000100000001001000100000011
00000000100000001010000100000011
00000000100000001100000100000011
00000000100000001101000100000011
00000000100000001000000100010011
00000000100000001010000100010011
00000000100000001011000100010011
00000000100000001100000100010011
00000000100000001110000100010011
00000000100000001000000100010011
00000000100000001001000100010011
00000000100000001101000100010011
01000000100000001101000100010011
00000000001000001000110000100011
00000000001000001001110000100011
00000000001000001010110000100011
00000000000000001111000010110111
00000000000000001111000010010111
00000010001000001000111001100011
00000010001000001001111001100011
00000010001000001100111001100011
00000010001000001101111001100011

00000010001000001110111001100011
00000010001000001111111001100011
00000001111000000000000011101111
00000000111100001000000101100111
```

结果分析

从测试结果可以看出汇编器能够将汇编指令编译成二进制结果，可以说明编写的汇编器可以完成实验要求，正确。通过本次实验，理解了汇编器的原理就在于定义好汇编指令集、二进制指令集，并且确定好二者之间的映射转换关系。