

（深圳）

实验报告

开课学期： 2022春季

课程名称：计算机组成原理（实验）

实验名称： 从C语言到机器码

实验性质： 综合设计型

实验学时： 2 地点： T2615

学生班级： 6班

学生学号： 200110619

学生姓名： 梁鑫嵘

作业成绩：

实验与创新实践教育中心制

2022年3月

|  |
| --- |
| 1、实验结果截图 |
|  |
| 2、汇编代码注释（只需写主程序和子程序即可） |
| // calc.c  *#include* <stdint.h>  *#include* <stdio.h>  int \_start() {    uint8\_t code = 19;    uint8\_t i = 0;    uint32\_t sum = 0;  *while* (i != 8) {  *if* (code & (1 << i)) sum += code << i;      i++;    }    uint32\_t sq = sum;    i = 0;    sum = 0;  *while* (i != 16) {  *if* (code & (1 << i)) sum += sq << i;      i++;    }  *// printf("%d\n", sum);*  *return* (int)sum;  }  // calc.s    .file "calc.c"        *// 指明编译的 .c 文件*    .option nopic         *// 编译器参数*    .attribute arch, "rv64i2p0\_m2p0\_a2p0\_f2p0\_d2p0\_c2p0"  *// 指令集*    .attribute unaligned\_access, 0    .attribute stack\_align, 16                            *// 堆内存对齐*    .text                                                 *// .text 程序段*    .align  1                                             *// 本程序段不需要对齐*    .globl  \_start                                        *// 将 \_start 段设置为全局可见*    .type \_start, @function                               *// 设置本段的属性为一个函数*  \_start:             *// \_start 段开始，程序从这里进入*    addi  sp,sp,-32   *// 将当前栈指针向低地址移动 32 个字节，表示在栈中新申请一个 8 个 word 大小的栈*    sd  s0,24(sp)     *// 在栈的第六个 word 处储存 s0 寄存器的值*    addi  s0,sp,32    *// 将帧缓存地址设置为栈底*    li  a5,19         *// 将 a5 寄存器的值设置为 19，即 `uint8\_t code = 19`*    sb  a5,-25(s0)    *// 将 a5(`code`)的低 8bit 保存到帧缓存地址向下第25字节*    sb  zero,-17(s0)  *// 将 0(`i`)的地 8bit 保存到帧缓存地址向下第17字节*    sw  zero,-24(s0)  *// 将 0(`sum`) 整个 word 保存到帧缓存地址向下第24字节*    j .L2             *// 跳转到标号 .L2*  .L4:                *// .L4 标号开始，为第一个 `while` 的循环体*    lbu a5,-25(s0)    *// 从帧缓存地址向下第 25 字节无符号取低字节到 a5，即 `a5 = code`*    sext.w  a4,a5     *// 将 a5 符号扩展的结果写入 a4，即 `a4 = SEXT(code)`，因为 `code` 是一个 uint8\_t 类型*    lbu a5,-17(s0)    *// 从帧缓存地址向下第 17 字节无符号取低字节到 a5，即 `a5 = i`*    sext.w  a5,a5     *// 将 a5 符号扩展的结果写入 a4，即 `a4 = SEXT(i)`，因为 `i` 是一个 uint8\_t 类型*    sraw  a5,a4,a5    *// a5 = a4 >>s a5，将 a4 算术右移 a5 位的结果写入 a5，即 `a5 = (i << i)`*    sext.w  a5,a5     *// 对 a5 做符号扩展*    andi  a5,a5,1     *// 将 a5(`i`) 与 1 相与存到 a5，即得到 `a5 = (1 << i)`*    sext.w  a5,a5     *// 对 a5 做符号扩展*    beq a5,zero,.L3   *// `if (code & (1 << i))`，不为0则跳转到 .L3*    lbu a5,-25(s0)    *// 从帧缓存地址向下第 25 字节取低字节到 a5，即 `a5 = code`*    sext.w  a4,a5     *// 对 a5 做符号扩展，写入 a4*    lbu a5,-17(s0)    *// 从帧缓存地址向下第 17 字节无符号取低字节到 a5，即 `a5 = i`*    sext.w  a5,a5     *// 对 a5 做符号扩展*    sllw  a5,a4,a5    *// a5 = a4 << a5，即将 a4 逻辑左移 a5 位的结果写入 a5，即 `a5 = code << i`*    sext.w  a5,a5     *// 对 a5 做符号扩展*    sext.w  a5,a5     *// 对 a5 做符号扩展*    lw  a4,-24(s0)    *// 从帧缓存地址向下第 24 字节取四字节到 a4，即 `a4 = sum`*    addw  a5,a4,a5    *// a5 = a4 + a5，即 `sum += code << i`*    sw  a5,-24(s0)    *// 将 a5 的值存入帧缓存向下第 24 字节处，即储存 `sum` 到缓存中*  .L3:                *// .L3 段，即第一个 `while` 循环体中 `i++` 部分*    lbu a5,-17(s0)    *// 从帧缓存地址向下第 17 字节无符号取低字节到 a5，即 `a5 = i`*    addiw a5,a5,1     *// a5 = a5 + 1，即 `a5 = i + 1`*    sb  a5,-17(s0)    *// 将 a5 的值低字节储存到帧缓存地址向下第 17 字节，即储存变量 `i`*  .L2:                *// .L2 标号位置*    lbu a5,-17(s0)    *// 从帧缓存向下第17字节加载半字到 a5 寄存器，即 a5 = `i`*    andi  a4,a5,0xff  *// 将加载出来的 a5 寄存器与 `0xff` 与，即取低 8bit 储存到 a4 寄存器*    li  a5,8          *// 将 a5 寄存器的值设置为 8，用于给循环设置终止条件*    bne a4,a5,.L4     *// `while (i != 8)`，不等于即跳转到 .L4，.L4 即循环体，等于就向下继续执行*    lw  a5,-24(s0)    *// 从帧缓存向下第 24 字节取四字节到 a5，即 `a5 = sum`*    sw  a5,-32(s0)    *// 将 a5(`sum`) 的值储存到帧缓存向下第 32 字节处，即 `sq = sum`*    sb  zero,-17(s0)  *// 将 0 的值储存到帧缓存向下第 17 字节处，即将缓存中变量 `i` 清零，即 `i = 0`*    sw  zero,-24(s0)  *// 将 0 的值储存到帧缓存向下第 24 字节处，即将缓存中变量 `sum` 清零，即 `sum = 0`*    j .L5             *// 跳转到 .L5 段*  .L7:                *// .L7 程序段，即第二个循环体*    lbu a5,-25(s0)    *// 从帧缓存向下第 25 字节无符号低字节加载到 a5，即 `a5 = code`*    sext.w  a4,a5     *// 将 a5 符号扩展结果写入 a4，即 `a4 = code`*    lbu a5,-17(s0)    *// 从帧缓存向下第 17 字节无符号低字节加载到 a5，即 `a5 = i`*    sext.w  a5,a5     *// 对 a5 做符号扩展*    sraw  a5,a4,a5    *// a5 = a4 >>s a5，将 a4 算术右移 a5 位的结果写入 a5，即 `a5 = (i << i)`*    sext.w  a5,a5     *// 对 a5 做符号扩展*    andi  a5,a5,1     *// 将 a5(`i`) 与 1 相与存到 a5，即得到 `a5 = (1 << i)`*    sext.w  a5,a5     *// 对 a5 做符号扩展*    beq a5,zero,.L6   *// `if (code & (1 << i))`，不为0则跳转到 .L6*    lbu a5,-17(s0)    *// 从帧缓存地址向下第 17 字节无符号取低字节到 a5，即 `a5 = i`*    sext.w  a5,a5     *// 对 a5 做符号扩展*    mv  a4,a5         *// 将 a4 的值设置为 a5，即 `a4 = a5 = i`*    lw  a5,-32(s0)    *// 从帧缓存地址向下第 32 字节取四字节到 a4，即 `a4 = sq`*    sllw  a5,a5,a4    *// a5 = a4 << a5，即将 a4 逻辑左移 a5 位的结果写入 a5，即 `a5 = sq << i`*    sext.w  a5,a5     *// 对 a5 做符号扩展*    lw  a4,-24(s0)    *// 从帧缓存地址向下第 24 字节取四字节到 a4，即 `a4 = sum`*    addw  a5,a4,a5    *// 将 a4 + a5 的值写入 a5，即 `sum += sq << i`*    sw  a5,-24(s0)    *// 将 a5 的值存入帧缓存向下第 24 字节处，即储存 `sum` 到缓存中*  .L6:                *// .L6 程序段，即第二个 `while` 循环体中 `i++` 部分*    lbu a5,-17(s0)    *// 从帧缓存地址向下第 17 字节无符号取低字节到 a5，即 `a5 = i`*    addiw a5,a5,1     *// a5 = a5 + 1，即 `a5 = i + 1`*    sb  a5,-17(s0)    *// 将 a5 的值低字节储存到帧缓存地址向下第 17 字节，即储存变量 `i`*  .L5:                *// .L5 程序段*    lbu a5,-17(s0)    *// 从帧缓存向下第 17 字节处无符号低字节加载到 a5，即 `a5 = i`*    andi  a4,a5,0xff  *// 将 a5 与 0xff 相与的结果写入 a4，即 `a4 = i & 0xff`*    li  a5,16         *// 将 a5 的值设置为 16，用于给循环设置终止条件*    bne a4,a5,.L7     *// `while (i != 16)`，不等于即跳转到 .L7*    lw  a5,-24(s0)    *// 从帧缓存向下第 24 字节加载四字节到 a5，即 `a5 = sum`*    mv  a0,a5         *// 将 a0 的值设置为 a5，即将 `sum` 作为返回值写入返回值寄存器*    ld  s0,24(sp)     *// 从栈底向上第 24 字节取八字节写入 s0，即恢复函数调用前的 s0 寄存器的值*    addi  sp,sp,32    *// sp 自增 32，即释放申请的 32 字节的栈空间*    jr  ra            *// 跳转并链接到上一次调用位置，即返回 `sum` 并跳转到上一调用帧*    .size \_start, .-\_start  *// 设置 \_start 段大小*    .ident  "GCC: (gca312387a) 10.2.0"  *// 写明编译器版本信息等* |
| 3、机器码注释（只需写主程序和子程序即可） |
| calc:     file format elf64-littleriscv  Disassembly of section .text:  00000000000100b0 <\_start>:    0: addi sp,sp,-32                     fe010113 rs1: sp, imm: -32, rd: sp    4: sd s0,24(sp)                       00813c23 rs1: sp, rs2: s0, imm: 24    8: addi s0,sp,32                      02010413 rs1: sp, imm: 32, rd: s0    c: li a5,19                           01300793 imm: 19, rd: a5   10: sb a5,-25(s0)                      fef403a3 rs1: s0, rs2: a5, imm: 6   14: sb zero,-17(s0)                    fe0407a3 rs1: s0, imm: 14   18: sw zero,-24(s0)                    fe042423 rs1: s0, imm: 8   1c: j 10128 <\_start+0x78>              05c0006f imm: 92(0x5c)   20: lbu a5,-25(s0)                     fe744783 rs1: s0, imm: -25, rd: a5   24: sext.w a4,a5                       0007871b rs1: a5, imm: 0, rd: a4   28: lbu a5,-17(s0)                     fef44783 rs1: s0, imm: -17, rd: a5   2c: sext.w a5,a5                       0007879b rs1: a5, imm: 0, rd: a5   30: sraw a5,a4,a5                      40f757bb rs1: a4, imm: 1039, rd: a5   34: sext.w a5,a5                       0007879b rs1: a5, imm: 0, rd: a5   38: andi a5,a5,1                       0017f793 rs1: a5, imm: 1, rd: a5   3c: sext.w a5,a5                       0007879b rs1: a5, imm: 0, rd: a5   40: beqz a5,1011c <\_start+0x6c>        02078663 rs1: a5, imm: 44(0x2c)   44: lbu a5,-25(s0)                     fe744783 rs1: s0, imm: -25, rd: a5   48: sext.w a4,a5                       0007871b rs1: a5, imm: 0, rd: a4   4c: lbu a5,-17(s0)                     fef44783 rs1: s0, imm: -17, rd: a5   50: sext.w a5,a5                       0007879b rs1: a5, imm: 0, rd: a5   54: sllw a5,a4,a5                      00f717bb rs1: a4, imm: 15, rd: a5   58: sext.w a5,a5                       0007879b rs1: a5, imm: 0, rd: a5   5c: sext.w a5,a5                       0007879b rs1: a5, imm: 0, rd: a5   60: lw a4,-24(s0)                      fe842703 rs1: s0, imm: -24, rd: a4   64: addw a5,a4,a5                      00f707bb rs1: a4, imm: 15, rd: a5   68: sw a5,-24(s0)                      fef42423 rs1: s0, rs2: a5, imm: 8   6c: lbu a5,-17(s0)                     fef44783 rs1: s0, imm: -17, rd: a5   70: addiw a5,a5,1                      0017879b rs1: a5, imm: 1, rd: a5   74: sb a5,-17(s0)                      fef407a3 rs1: s0, rs2: a5, imm: 14   78: lbu a5,-17(s0)                     fef44783 rs1: s0, imm: -17, rd: a5   7c: andi a4,a5,255                     0ff7f713 rs1: a5, imm: 255, rd: a4   80: li a5,8                            00800793 imm: 8, rd: a5   84: bne a4,a5,100d0 <\_start+0x20>      f8f71ee3 rs1: a4, rs2: a5, imm: -100(0xffffff9c)   88: lw a5,-24(s0)                      fe842783 rs1: s0, imm: -24, rd: a5   8c: sw a5,-32(s0)                      fef42023 rs1: s0, rs2: a5, imm: 0   90: sb zero,-17(s0)                    fe0407a3 rs1: s0, imm: 14   94: sw zero,-24(s0)                    fe042423 rs1: s0, imm: 8   98: j 101a0 <\_start+0xf0>              0580006f imm: 88(0x58)   9c: lbu a5,-25(s0)                     fe744783 rs1: s0, imm: -25, rd: a5   a0: sext.w a4,a5                       0007871b rs1: a5, imm: 0, rd: a4   a4: lbu a5,-17(s0)                     fef44783 rs1: s0, imm: -17, rd: a5   a8: sext.w a5,a5                       0007879b rs1: a5, imm: 0, rd: a5   ac: sraw a5,a4,a5                      40f757bb rs1: a4, imm: 1039, rd: a5   b0: sext.w a5,a5                       0007879b rs1: a5, imm: 0, rd: a5   b4: andi a5,a5,1                       0017f793 rs1: a5, imm: 1, rd: a5   b8: sext.w a5,a5                       0007879b rs1: a5, imm: 0, rd: a5   bc: beqz a5,10194 <\_start+0xe4>        02078463 rs1: a5, imm: 40(0x28)   c0: lbu a5,-17(s0)                     fef44783 rs1: s0, imm: -17, rd: a5   c4: sext.w a5,a5                       0007879b rs1: a5, imm: 0, rd: a5   c8: mv a4,a5                           00078713 rs1: a5, imm: 0, rd: a4   cc: lw a5,-32(s0)                      fe042783 rs1: s0, imm: -32, rd: a5   d0: sllw a5,a5,a4                      00e797bb rs1: a5, imm: 14, rd: a5   d4: sext.w a5,a5                       0007879b rs1: a5, imm: 0, rd: a5   d8: lw a4,-24(s0)                      fe842703 rs1: s0, imm: -24, rd: a4   dc: addw a5,a4,a5                      00f707bb rs1: a4, imm: 15, rd: a5   e0: sw a5,-24(s0)                      fef42423 rs1: s0, rs2: a5, imm: 8   e4: lbu a5,-17(s0)                     fef44783 rs1: s0, imm: -17, rd: a5   e8: addiw a5,a5,1                      0017879b rs1: a5, imm: 1, rd: a5   ec: sb a5,-17(s0)                      fef407a3 rs1: s0, rs2: a5, imm: 14   f0: lbu a5,-17(s0)                     fef44783 rs1: s0, imm: -17, rd: a5   f4: andi a4,a5,255                     0ff7f713 rs1: a5, imm: 255, rd: a4   f8: li a5,16                           01000793 imm: 16, rd: a5   fc: bne a4,a5,1014c <\_start+0x9c>      faf710e3 rs1: a4, rs2: a5, imm: -96(0xffffffa0)  100: lw a5,-24(s0)                      fe842783 rs1: s0, imm: -24, rd: a5  104: mv a0,a5                           00078513 rs1: a5, imm: 0, rd: a0  108: ld s0,24(sp)                       01813403 rs1: sp, imm: 24, rd: s0  10c: addi sp,sp,32                      02010113 rs1: sp, imm: 32, rd: sp  110: ret                                00008067 rs1: ra, imm: 0 |

**说明：**

1. 在使用 spike 运行程序的时候，代码调用了 printf 函数；为了简化反编译内容，在运行中使用的程序使用 \_start 作为入口，并且将结果直接作为返回值返回。

运行时使用的程序代码

// calc\_print.c

*#include* <stdint.h>

*#include* <stdio.h>

int main() {

  uint8\_t code = 19;

  uint8\_t i = 0;

  uint32\_t sum = 0;

*while* (i != 8) {

*if* (code & (1 << i)) sum += code << i;

    i++;

  }

  uint32\_t sq = sum;

  i = 0;

  sum = 0;

*while* (i != 16) {

*if* (code & (1 << i)) sum += sq << i;

    i++;

  }

  printf("%d\n", sum);

*return* 0;

}

// hello\_print.c

*#include* <stdio.h>

int main() {

    printf("Hello World!\n");

*return* 0;

}

1. calc程序实现的是学号的立方，即 19^3 = 6859
2. 解码反编译指令，我使用的是C语言和Python语言进行的解码，代码如下
   1. decoder.c

*#include* <stdint.h>

*#include* <stdio.h>

*#include* <stdlib.h>

*#define* concat\_temp(x, y) x##y

*#define* concat(x, y) concat\_temp(x, y)

*#define* concat3(x, y, z) concat(concat(x, y), z)

*#define* concat4(x, y, z, w) concat3(concat(x, y), z, w)

*#define* concat5(x, y, z, v, w) concat4(concat(x, y), z, v, w)

*#define* SEXT(x, len)     \

  ({                     \

    struct {             \

      int64\_t n : (len); \

    } \_\_x = {.n = x};    \

    (int64\_t) \_\_x.n;     \

  })

struct lookup\_result\_t {

  uint32\_t imm;

  uint32\_t rs1;

  uint32\_t rs2;

  uint32\_t rd;

  uint32\_t fun3;

  uint32\_t fun7;

};

typedef struct lookup\_result\_t lookup\_result;

lookup\_result lookup\_error = {

    .imm = 0, .rs1 = 0, .rs2 = 0, .rd = 0, .fun3 = 0, .fun7 = 0};

int lookup\_table(const char \*string, uint32\_t inst) {

  const char \*p = string;

  int skip = 0;

  int ok = 1;

*while* (p && \*p) {

*if* (\*p == '?') {

      p++;

*continue*;

    } *else* *if* (\*p == ' ') {

      p++;

      skip++;

*continue*;

    } *else* *if* ((((inst << (uint32\_t)(p - string - skip)) & 0x80000000)

                    ? 1

                    : 0) == (\*p - '0')) {

      p++;

*continue*;

    } *else* {

      ok = 0;

*break*;

    }

  }

*return* ok;

}

lookup\_result t\_I(uint32\_t i) {

  lookup\_result r = {.imm = SEXT(i >> 20, 12),

                     .rs1 = (i >> 15) & 0x1f,

                     .rs2 = 0,

                     .rd = (i >> 7) & 0x1f,

                     .fun3 = (i >> 12) & 0x3,

                     .fun7 = 0};

*return* r;

}

lookup\_result t\_R(uint32\_t i) {

  lookup\_result r = {.imm = 0,

                     .rs1 = (i >> 15) & 0x1f,

                     .rs2 = (i >> 20) & 0x1f,

                     .rd = (i >> 7) & 0x1f,

                     .fun3 = (i >> 12) & 0x3,

                     .fun7 = (i >> 25) & 0x7f};

*return* r;

}

lookup\_result t\_S(uint32\_t i) {

  lookup\_result r = {.imm = (i >> 7) & 0x1f + SEXT((i >> 25), 7),

                     .rs1 = (i >> 15) & 0x1f,

                     .rs2 = (i >> 20) & 0x1f,

                     .rd = 0,

                     .fun3 = (i >> 12) & 0x3,

                     .fun7 = 0};

*return* r;

}

lookup\_result t\_B(uint32\_t i) {

  lookup\_result r = {

      .imm = SEXT((((i >> 8) & 0xf) << 1) + (((i >> 25) & 0x3f) << 5) +

                      (((i & 0x80) ? 1 : 0) << 11) +

                      (((i & 0x80000000) ? 1 : 0) << 12),

                  12),

      .rs1 = (i >> 15) & 0x1f,

      .rs2 = (i >> 20) & 0x1f,

      .rd = 0,

      .fun3 = (i >> 12) & 0x3,

      .fun7 = 0};

*return* r;

}

lookup\_result t\_U(uint32\_t i) {

  lookup\_result r = {.imm = SEXT(i >> 12, 20),

                     .rs1 = 0,

                     .rs2 = 0,

                     .rd = (i >> 7) & 0x1f,

                     .fun3 = 0,

                     .fun7 = 0};

*return* r;

}

lookup\_result t\_J(uint32\_t i) {

  lookup\_result r = {.imm = SEXT((((i >> 21) & 0x3ff) << 1) + (((i & 0x80000) ? 1 : 0) << 11) +

                                     (((i >> 12) & 0xff) << 12) +

                                     (((i & 0x80000000) ? 1 : 0) >> 12),

                                 20),

                     .rs1 = 0,

                     .rs2 = 0,

                     .rd = (i >> 7) & 0x1f,

                     .fun3 = 0,

                     .fun7 = 0};

*return* r;

}

const char \*r\_regs[] = {"$0", "ra", "sp",  "gp",  "tp", "t0", "t1", "t2",

                        "s0", "s1", "a0",  "a1",  "a2", "a3", "a4", "a5",

                        "a6", "a7", "s2",  "s3",  "s4", "s5", "s6", "s7",

                        "s8", "s9", "s10", "s11", "t3", "t4", "t5", "t6"};

char \*disp(uint32\_t inst, lookup\_result r) {

  char \*buf = malloc(128);

  char \*p = buf;

  sprintf(p, "inst: %08x ", inst);

  p += 15;

*if* (r.rs1) {

    sprintf(p, "rs1: %-3s ", r\_regs[r.rs1]);

    p += 9;

  }

*if* (r.rs2) {

    sprintf(p, "rs2: %-3s ", r\_regs[r.rs2]);

    p += 9;

  }

  sprintf(p, "imm: %08x ", r.imm);

  p += 14;

*if* (r.rd) {

    sprintf(p, "rd: %-3s ", r\_regs[r.rd]);

    p += 8;

  }

  \*p = '\0';

*return* buf;

}

*#define* lookup(string, inst, type)    \

*do* {                                \

*if* (lookup\_table(string, inst)) { \

*return* concat(t\_, type)(inst);  \

    }                                 \

  } *while* (0)

lookup\_result lookup\_instr(uint32\_t inst) {

  lookup("????????????????? ??? ????? 0000011", inst, I);

  lookup("????????????????? ??? ????? 0010011", inst, I);

  lookup("????????????????? ??? ????? 0011011", inst, I);

  lookup("????????????????? ??? ????? 0111011", inst, I);

  lookup("????????????????? ??? ????? 0100011", inst, S);

  lookup("????????????????? ??? ????? 0110011", inst, R);

  lookup("????????????????? ??? ????? 0010011", inst, R);

  lookup("????????????????? ??? ????? 1100011", inst, B);

  lookup("????????????????? ??? ????? 1101111", inst, J);

  lookup("????????????????? ??? ????? 1100111", inst, I);

*return* lookup\_error;

}

int is\_lookup\_err(lookup\_result res) {

*return* res.imm == lookup\_error.imm && res.rs1 == lookup\_error.rs1 &&

         res.rs2 == lookup\_error.rs2 && res.rd == lookup\_error.rd &&

         res.fun3 == lookup\_error.fun3 && res.fun7 == lookup\_error.fun7;

}

*#define* rAssert(x, string)         \

*do* {                             \

*if* (!(x)) {                    \

      printf("Error: %s", string); \

*return* 1;                    \

    }                              \

  } *while* (0)

int main(int argc, char \*\*argv) {

  uint32\_t inst = argc == 2 ? (uint32\_t)atoi(argv[1]) : 0x01430513;

  lookup\_result res = lookup\_instr(inst);

*if* (is\_lookup\_err(res)) {

    printf("Error inst: %08x", inst);

  } *else* {

    printf("%s", disp(inst, res));

  }

*return* 0;

}

* 1. get\_instr.py

*import* os

index = 0

*with* os.popen("riscv64-linux-gnu-objdump -d calc") *as* f:

  lines = f.readlines()

*# lines = [line for line in lines if len(line) > 20 and line[8] == ':']*

*# lines = [line.split(':')[-1].strip() for line in lines]*

*for* line *in* lines:

*if* len(line.strip()) == 0:

*continue*

*if* not (len(line) > 20 and line[8] == ':'):

      print(line)

*continue*

    line = line.split(':')[-1].strip()

    items = [item.strip() *for* item *in* line.split('\t')]

    data = {

      'inst': int(items[0], 16),

      'code': ' '.join(items[1:])

    }

*with* os.popen(f"./decoder {data['inst']}") *as* r:

      decoded = r.readlines()[0]

*if* decoded[0] == 'E':

        data['decode'] = None

*else*:

        de = [d *for* d *in* decoded.replace(':', '').split(' ')[2:] *if* len(d) > 0]

        info = {de[i]: de[i+1] *for* i *in* range(0, len(de), 2)}

        info['imm'] = int(info['imm'], 16)

        imm\_raw = info['imm']

*if* info['imm'] & 0x80000000:

          info['imm'] = info['imm'] - 0xffffffff - 1

*if* '0x' in data['code']:

          info['imm'] = f"{info['imm']}({hex(imm\_raw)})"

        data['decode'] = info

    print("%3x: " % index + data['code'] + '\t' \* (5 - (len(data['code']) + 6) // 8) + f"{data['inst']:08x} " + str(data['decode']).replace('{', '').replace('}', '').replace("'", ''))

    index += 4

1. 其他代码
   1. build.mk

*# cross-compiler settings*

*# CROSS\_COMPILE := riscv64-linux-gnu-*

CROSS\_COMPILE := riscv64-unknown-elf-

AS        = $(CROSS\_COMPILE)gcc

CC        = $(CROSS\_COMPILE)gcc

CXX       = $(CROSS\_COMPILE)g++

LD        = $(CROSS\_COMPILE)ld

OBJDUMP   = $(CROSS\_COMPILE)objdump

OBJCOPY   = $(CROSS\_COMPILE)objcopy

READELF   = $(CROSS\_COMPILE)readelf

RISCV     = $(HOME)/riscv

PK        = $(RISCV)/riscv64-linux-gnu/bin/pk

SPIKE     = spike

* 1. Makefile

*include* ../scripts/build.mk

ALL := hello calc hello.bin calc.bin calc.s calc.o calc.i decoder

CFLAGS := -static -nostdlib -march=rv64i -mabi=lp64

all: $(ALL)

%.i: %.c

  $(CC) $(CFLAGS) -E $< -o $@

%.s: %.i

  $(CC) $(CFLAGS) -S $< -o $@

%.o: %.s

  $(CC) $(CFLAGS) -c $< -o $@

%: %.o

  $(CC) $(CFLAGS) $< -o $@

%\_dump.txt: %.o

  $(OBJDUMP) -j .text -d $< > $@

%.txt: %

  $(OBJDUMP) -S $< > $@

%\_print: %

  $(CC) -o $@ $@.c

%.bin: %

  $(OBJCOPY) -O binary $< $@

decoder: decoder.c

  gcc -o decoder decoder.c

clean:

*-*rm -rf \*.s \*.i \*.o \*.txt $(ALL)

run: $(ALL) calc\_print hello\_print

  $(SPIKE) $(PK) hello\_print

  $(SPIKE) $(PK) calc\_print

.PHONY: all clean