Lab2

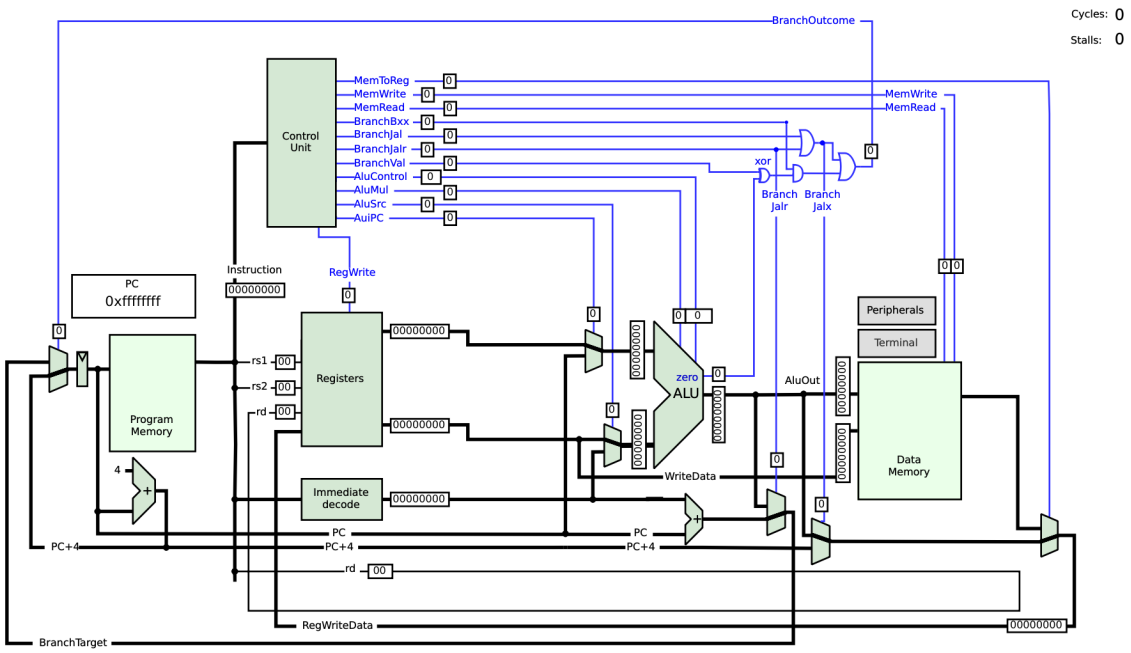
CS2305 Computer System Architecture

April 7, 2024

0 Requirements

- 最终需要提交一个 PDF 文档，命名方式为 “Lab2_ 学号 _ 姓名.pdf”。pdf 文件首页为封面，注明 “Lab2”，学号，姓名，专业等个人信息，第 2 页开始答题，无需抄题，但需注明题号。
- 对于思考题，体现你的探索和思考，言之有理即可，不得抄袭，表述尽可能精简直接，允许中文作答。

1 Assignments



1. 上图是QtRvSim，一个 RISCV 模拟器中绘制的 RISCV 处理器结构图。请简要描述图中各个部分的名称与功能，然后结合各个组件描述一条指令执行的取指、译码、执行、访存、写回五个阶段的行为。

2. 下面是一个简单的 C 语言程序：

```c
int add(int a, int b)
{
    return a + b;
}

int main()
{
    int result = add(1, 2);
    return 0;
}
```

使用`riscv64-unknown-elf-gcc`编译后，得到的汇编代码为：

```
1  add:
2      addi    sp,sp,-32
3      sw  s0,28(sp)
4      addi    s0,sp,32
5      sw  a0,-20(s0)
6      sw  a1,-24(s0)
7      lw  a4,-20(s0)
8      lw  a5,-24(s0)
9      add a5,a4,a5
10     mv  a0,a5
11     lw  s0,28(sp)
12     addi    sp,sp,32
13     jr  ra
14 main:
15     addi    sp,sp,-32
16     sw  ra,28(sp)
17     sw  s0,24(sp)
18     addi    s0,sp,32
19     li  a1,2
20     li  a0,1
21     call    add
22     sw  a0,-20(s0)
23     li  a5,0
24     mv  a0,a5
25     lw  ra,28(sp)
26     lw  s0,24(sp)
27     addi    sp,sp,32
28     jr  ra
```

请指出 C 语言中a+b对应哪几行汇编代码？逐行解释它们分别执行了什么操作，并指出这些操作涉及到图 1 中的哪些组件。

3. 使用前文提到的 QtRvSim 模拟器执行上述程序（参考教程见后），描述执行汇编语句add（第 9 行）时，RISC-V 处理器的状态，如 PC 值、指令译码结果、ALU 的输入输出等。并解释代码的执行过程。

2 Attachments

2.1 关于 RISC-V 指令集

请参考 https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf，https://zhuanlan.zhihu.com/p/671680439，https://msyksphinz-self.github.io/riscv-isadoc/html/rvi.html。

第一个链接是 RISC-V 官方的指令集手册，内容详细但是阅读有一定门槛；第二个链接是知乎上对 RISC-V 指令集的一个简单介绍；第三个链接中包含了所有 RISC-V 指令的详细介绍。

2.2 关于第二题中的程序

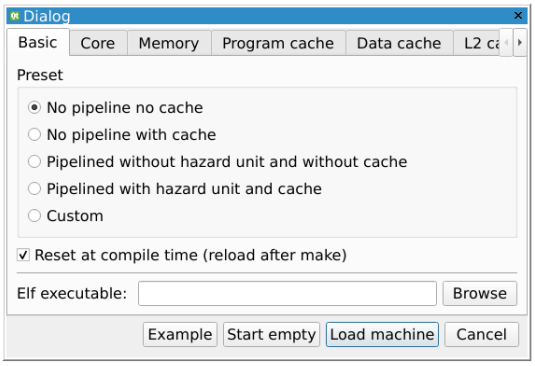
main部分解释如下：

- `addi sp,sp,-32`：将栈指针 `sp` 减去 32，为局部变量和保存的寄存器分配栈空间。
- `sw ra,28(sp)`：将返回地址寄存器 `ra` 保存到栈上，偏移量为 28。这是为了在函数调用后能够返回到正确的地址。
- `sw s0,24(sp)`：将帧指针 `s0`（保存了上一个栈帧的基地址）保存到栈上，偏移量为 24。
- `addi s0,sp,32`：将当前栈帧的底部地址（即调整后的栈指针 `sp`）加上 32，设置为新的帧指针 `s0`。这是为了后续指令能够基于 `s0` 定位局部变量和保存的寄存器。
- `li a1,2` 和 `li a0,1`：准备 `add` 函数的参数。
- `call add`：调用 `add` 函数。`call` 是一个宏，实际上会被展开为 `jal` 指令，跳转到 `add` 函数的地址，并将返回地址（即 `call` 指令后面的地址）保存到 `ra` 寄存器。
- `sw a0,-20(s0)`：将 `add` 函数的返回值（保存在 `a0` 中）存储到栈帧中，偏移量为-20。
- `li a5,0` 和 `mv a0,a5`：设置程序的退出状态。
- `lw ra,28(sp)` 和 `lw s0,24(sp)`：从栈上恢复寄存器的值。
- `addi sp,sp,32` 和 `jr ra`：释放栈空间并返回到调用者。

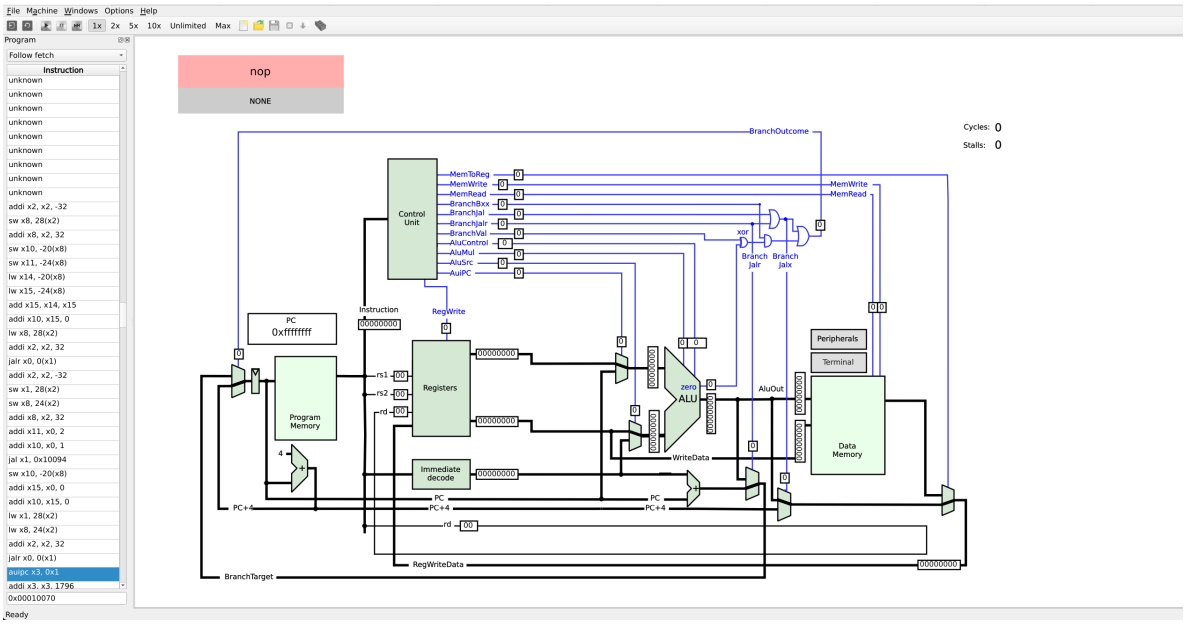
在解释add函数时可以参考。

2.3 关于 QtRvSim

QtRVSim 是一个基于 Qt 框架开发的 RISC-V 模拟器。它被设计来模拟 RISC-V 指令集架构（ISA）的处理器，提供一个图形用户界面（GUI），通过这个界面，用户可以方便地加载、执行 RISC-V 汇编或二进制代码，并实时观察程序的执行情况，包括寄存器的值、内存状态和指令执行的步骤。QtRVSim 可以在 https://comparch.edu.cvut.cz/qtrvsim/app/访问。github 地址为 https://github.com/cvut/qtrvsim。要完成此次实验，你需要在 QtRVSim 中加载此次作业给出的程序，方法如下：



在上面的页面中选择”Browse”，然后在弹出的窗口中选择作业中给出的编译好的程序文件Lab2，然后点击 Load Machine。如果加载成功，页面如下：



上述按钮效果分别为加载新模拟（重新配置并加载程序）、重新开始本次模拟（清零内存、寄存器，并让程序回到开始位置）、开始执行（程序将自动执行，执行指令的间隔由后面的倍数控制）、暂停执行和单步执行（执行下一条指令）。为了方便观察，建议使用单步执行。