

课程项目 Y86模拟器

Deadline: 2023-12-20 23:59:59

项目概述

根据CSAPP中对Y86指令集的描述和讲解，自行实现一个任意架构的CPU处理器。

项目要求

- 项目完成人数1~2人。其中单人完成的同学可以适当降低一些工作量的要求。多人组队的同学需要把参与项目的每个人的**工作量和贡献度**都写在报告中。
- 支持CSAPP中Y86的**基本指令**
- **程序输入输出满足基本格式，输出与标准答案一致**
- 需要提交项目报告和项目文件
- 对技术栈不做限制，可以自由选择编程语言(python,c++,SystemVerilog等)、操作系统
- 在满足项目要求的条件下，有能力的同学可以做额外的完善，例如更好的架构设计、更丰富的指令、所学知识的综合等,即添加自己的亮点
- 要求在期末之前进行课堂汇报，包括项目设计的亮点等

分值分配

- 基础功能(80%)
- 项目展示与报告(15%)
- 项目亮点(5%)

项目内容

在项目中，你需要设计实际一个cpu模拟器，你的程序将从 `stdio` 读取机器码 `.yo` 文件，然后在 `stdout` 按要求输出初始状态和每条指令执行后的CPU状态的日志。（回想bomblab的重定向）

最后的运行方式类似如下：

```
$ ./cpu < example.yo > example.json
或
$ python cpu.py < example.yo > example.yaml
```

模拟器输入

请以文件包给出的包含了机器码和汇编码的 `.yo` 文件作为模拟器输入，自行编写代码处理
样例如下：

```
0x00a: 30f23f00000000000000 | irmovq $63, %rdx # src and dst have 63 elements
0x014: 30f69802000000000000 | irmovq dest, %rsi # dst array
0x01e: 30f79000000000000000 | irmovq src, %rdi # src array
```

模拟器输出

- 输出一份 json 格式文件或 yaml 格式文件，可使用库或自行编写代码
- 要求在每条指令执行完毕后输出完整的寄存器信息和内存非零值(八字节对齐，按小端法解释为十进制有符号整数)。内存非零值指{(内存地址,内存值)|内存值 \neq 0, \forall 内存地址}，即所有非零内存值的内存地址-值键值对。
- 所有输出(含内存地址、寄存器值、内存值)均以十进制输出
- 最终完整输出样例如下，无需担心每次log内key-value的排列顺序，但要确保列表内log的顺序与程序执行顺序一致

- json

```
[
  {
    "PC": 0,
    "REG": {
      "rax": 1,
      "rcx": -2,
      "rdx": 3,
    },
    "CC": {
      "ZF": 0,
      "SF": 0,
      "OF": 0
    },
    "STAT": 1,
    "MEM": {
      "64": 4294901760,
      "72": 65535,
      ...
    }
  },
  ...
]
```

- yaml

```
- PC: 0
  REG:
    rax: 1
    rcx: -2
    rdx: 3
    ...
  CC:
    ZF: 0
    SF: 0
    OF: 0
  MEM:
    64: 4294901760
    72: 65535
    ...
  STAT: 1
- ...
```

实现前端界面(可选)

为了方便期末pre和行CPU运行期间相关信息的展示，同学们可以选择性完成前端界面。**前端页面不要求强制完成，完成不能获得额外分数**，你甚至可以用命令行作为前端页面。前端界面的时间方法非常多，这里不做限制，助教可以推荐几个常用的技术思路：

桌面应用

思路是不输出此时处理器的状态，直接显示到前端

- QT库：一个非常成熟的跨系统GUI库，有C++, Python等语言的API，网上也能找到很多资料
- GTK库：另一个非常成熟且很有开源精神的GUI库，提供了C, JS, Python等语言的API [The GTK Project - A free and open-source cross-platform widget toolkit](#)
- Electron：一个用写网页的方法写GUI的库，如果你有web基础，可以考虑用这个[Electron | Buildcross-platform desktop apps with JavaScript, HTML, and CSS. \(electronjs.org\)](#)
- UWP, SwiftUI.....

web应用

思路是后端提供API的Web接口，前端读取Web接口的数据显示

P.S. 你可以直接传上一阶段生成的json数据

- Django&Flask+HTML/JS：非常成熟的Python Web框架，熟练的话半天时间就能把整个PJ写完
- NodeJS+HTML：非常成熟的JS后端，性能据说也不错
- Go/Java/...+HTML

不知道从哪里开始？

- 阅读CSAPP，了解Y86指令集
- 先试着读取文件内容和解析文件内容为一条条对应指令
- 画图设计你的每条指令的执行过程，回想课上讲的cpu设计
- 先从简单的一条指令开始，逐步增加指令
- 在每条指令结束后记得保存寄存器和内存的状态（变量状态）

什么是项目亮点

项目亮点是指你在完成基本功能的基础上，自行添加的功能或者设计，我们可以给出如下几个方向的建议：

- 任何你认为你所作的巧妙独特的地方
- 更好的cpu架构设计
- 更丰富的指令集
- 更好的展示效果
- 如何结合我们之前学过的实验内容丰富功能

项目展示

项目展示将在课程最后一到两周进行，具体时间待定。希望同学们能尽早完成pj方便课上展示。展示形式为每组5分钟左右的汇报，需要同学准备相应的ppt，汇报内容可以包括但不限于：

- cpu架构设计
- cpu实机运行效果
- 项目亮点

我们的pre的分数**全部**由投票打分得到

提交方式

请将项目报告、项目文件打包提交，命名格式为： 姓名_学号_PJ

- 项目报告内需包含你的CPU的设计，前端的设计（如果有），代码运行的方法。在讲清楚的情况下越短越好，以及你的意见+建议（可选）。
- 项目文件内需包含你所有的代码和静态文件(如前端的图片，设计文档等)
- 请**不要**将.git或者二进制文件附在压缩包里

注意事项

- 本次项目旨在考查同学们对Y86指令集和基本处理器设计思想的掌握程度，满足项目要求即可获得大部分分数，请自行平衡好时间
- 禁止抄袭代码，鼓励自行学习相关知识丰富模拟器功能
- 小组如果中途发生变动，需要向助教说明情况