

title: Final PJ
author: 胥昊天

date: 2024-11-25

Deadline: 2024-12-26 23:59:59

一、项目简介

CPU架构相关实验。

本项目旨在帮助大家理解基本的处理器架构和指令执行机制，为后续学习计算机组成和体系结构打下基础。

项目的核心目标是：

1. **实现Y86处理器**：根据Y86指令集规范，构建一个简单的CPU处理器模型。
2. **测试正确性**：运行测试程序，验证处理器对Y86指令集的支持和执行的正确性。

二、项目要求

- **项目1~2人完成**。双人完成得分上限为90%，单人完成得分上限为100%。
- 设计的CPU需要支持Y86的**基本指令**。
- 输入输出符合格式，通过测试程序。
- 提交实验代码和报告，简要地描述你的设计思路。
- **对CPU架构不做限制**，单周期与流水线架构都是OK的。
- **对技术栈不做限制**，可以自由选择编程语言（python, c++, verilog.....）。
- **对前端界面不做要求**。
- 满足以上要求，有能力和兴趣的同学可以做额外的完善，如：升级架构、丰富指令集。
- 期末之前，完成课堂汇报。

三、项目提示

CPU执行大致分为6个阶段：

- **取指 (Fetch)**：取指阶段从内存中读取指令的字节，使用程序计数器（PC）作为内存地址。
- **译码 (Decode)**：从寄存器中读取用于运算的数据。
- **执行 (Execute)**：算术/逻辑单元（ALU）执行当前指令，或执行算术运算或计算内存引用的有效地址，或增加或减少栈指针。其间可能会设置条件码（condition code）。
- **访存 (Memory)**：可能会将数据写入内存，或者从内存中读取数据。
- **写回 (Write back)**：将最多两个数据写入寄存器。
- **更新PC (PC update)**：将PC更新为下一个指令的内存地址。

一个顺序执行的 Y86 CPU 架构图可能如下：

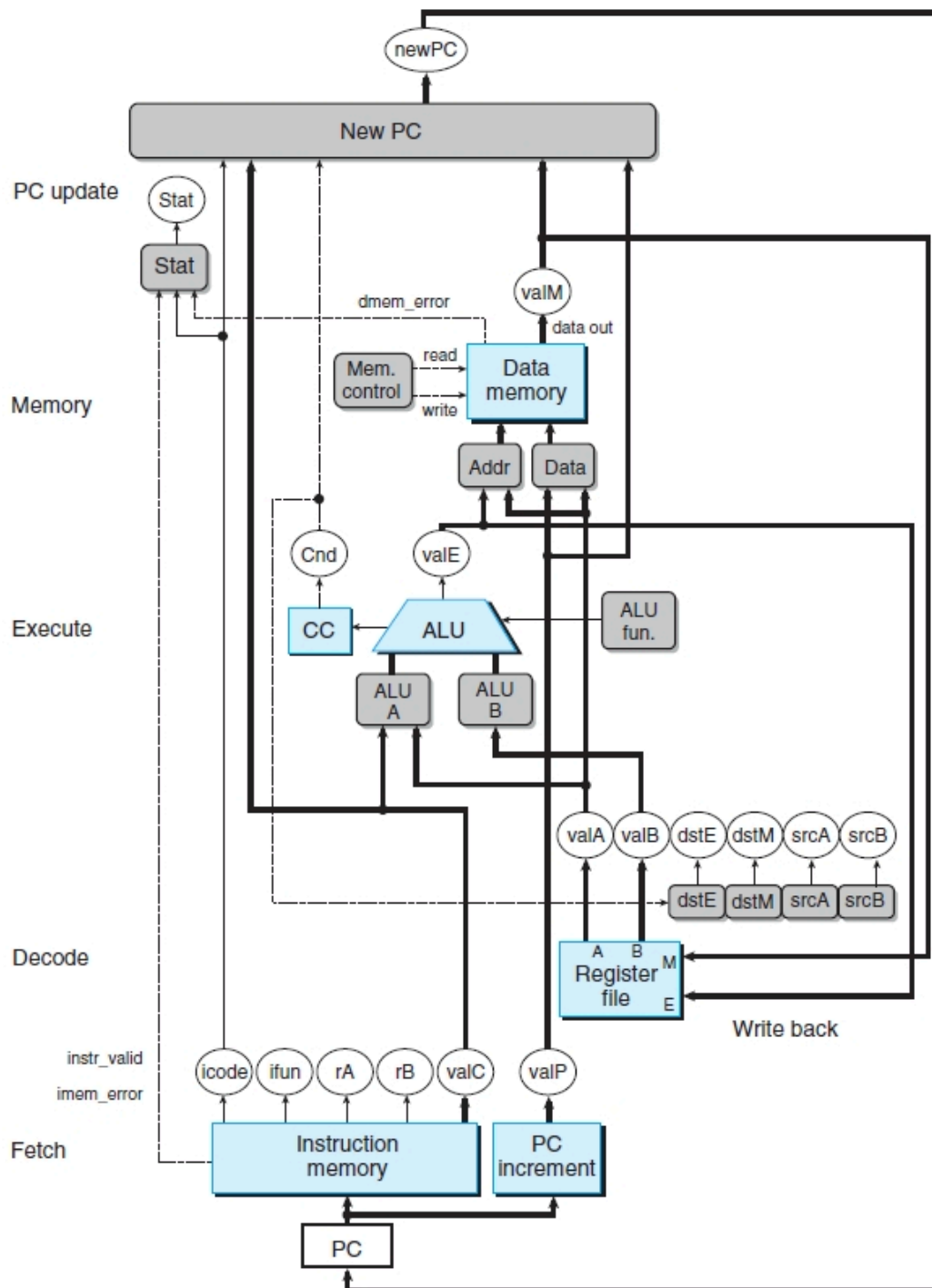


Figure 4.23 Hardware structure of SEQ, a sequential implementation. Some of the control signals, as well as the register and control word connections, are not shown.

Y86指令集在CSAPP书中第四章有详细的介绍。以下是 Y86 指令集的概述图。

CSCI 370: Computer Architecture

Y86-64 Reference

Instruction Format

halt	0 0	
nop	1 0	
rrmovq rA , rB	2 0 rA rB	
cmovXX rA , rB	2 fn rA rB	
irmovq V , rB	3 0 F rB V	
rmmovq rA , D(rB)	4 0 rA rB D	
mrmovq D(rB) , rA	5 0 rA rB D	
addq rA , rB	6 0 rA rB	
subq rA , rB	6 1 rA rB	
andq rA , rB	6 2 rA rB	
xorq rA , rB	6 3 rA rB	
jmp Dest	7 0 Dest	
jXX Dest	7 fn Dest	
call Dest	8 0 Dest	
ret	9 0	
pushq rA	A 0 rA F	
popq rA	B 0 rA F	

fn Codes
1 le 3 e 5 ge
2 l 4 ne 6 g

Registers

ID	Enc	Usage	
%rdi	7	arg1	caller-saved
%rsi	6	arg2	
%rdx	2	arg3	
%rcx	1	arg4	
%r8	8	arg5	
%r9	9	arg6	
%rax	0	return	callee-saved
%r10	A	general	
%r11	B	general	
%rbx	3	general	
%r12	C	general	
%r13	D	general	
%r14	E	general	
%rsp	4	stack ptr	
%rbp	5	base ptr	
	F	no reg	

Status Conditions

AOK	1	Normal
HLT	2	Halt Encountered
ADR	3	Bad Address
INS	4	Invalid Instruction

HCL Y86-64 Hardware Registers

stage	register(s)	description
Fetch	icode, ifun	Read instruction byte
	rA, rB	Read register byte
	valC	Read constant word
	valP	Compute next PC
Decode	valA, srcA	Read operand A
	valB, srcB	Read operand B
Execute	valE	Perform ALU operation
	cnd	Set/Use Condition Code
Memory	valM	Memory Read/Write
Writeback	dstE	Write back ALU result
	dstM	Write back Mem result
PC Update	PC	Update PC

Y86-64 Data Example

```
.align 8
Array:
.quad 0x0000000000000001
.quad 0x0000000000000002
.quad 0x0000000000000003
.quad 0x0000000000000004
```

Assembly Translation Example

```
/* find number of elements in null-terminated list */
long len(long* a) {
    long len;
    for (len = 0; a[len]; ++len)
        ;
    return len;
}

len:
    irmovq $1, %r8          # Constant 1
    irmovq $8, %r9          # Constant 8
    irmovq $0, %rax         # len = 0
    mrmovq (%rdi), %rdx     # val = *a
    andq %rdx, %rdx         # Test val
    je Done                 # If zero, goto Done

Loop:
    addq %r8, %rax          # len++
    addq %r9, %rdi          # a++
    mrmovq (%rdi), %rdx     # val = *a
    andq %rdx, %rdx         # Test val
    jne Loop               # If !0, goto Loop

Done:
    ret
```

为了方便期末pre和CPU运行期间相关信息的展示，同学们可以选择完成前端界面。前端页面不要求强制完成，完成不能获得额外分数。这里我们提供一些指南，供感兴趣的同学学习。

- 桌面应用：
 - QT库：一个非常成熟的跨系统 GUI 库，有 C++，Python 等语言的 API。
 - 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
 - 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。
- web应用：
 - 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，性能也不错
 - Go/Java/... + HTML

四、项目测试

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

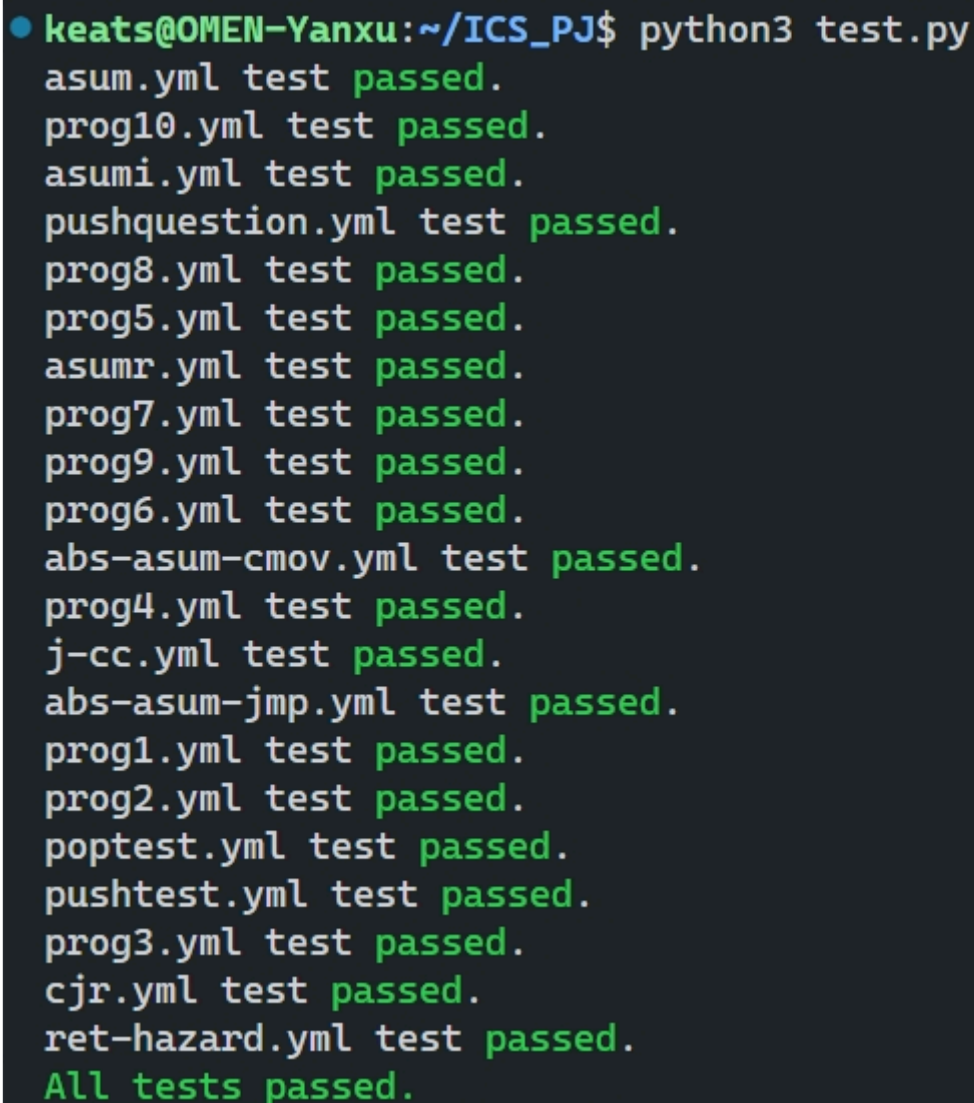
你的程序可能的运行方式如下：

```
$ ./cpu < test/example.yo > output/example.yml
```

```
$ python cpu.py < test/example.yo > output/example.yml
```

我们提供了测试脚本 `test.py`，在工作目录下运行测试脚本：

```
$ python test.py
```



```
● keats@OMEN-Yanxu:~/ICS_PJ$ python3 test.py
asum.yml test passed.
prog10.yml test passed.
asumi.yml test passed.
pushquestion.yml test passed.
prog8.yml test passed.
prog5.yml test passed.
asumr.yml test passed.
prog7.yml test passed.
prog9.yml test passed.
prog6.yml test passed.
abs-asum-cmov.yml test passed.
prog4.yml test passed.
j-cc.yml test passed.
abs-asum-jmp.yml test passed.
prog1.yml test passed.
prog2.yml test passed.
poptest.yml test passed.
pushtest.yml test passed.
prog3.yml test passed.
cjr.yml test passed.
ret-hazard.yml test passed.
All tests passed.
```

输入格式

请以 `test` 目录下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: 30f23f0000000000000000 | irmovq $63, %rdx # src and dst have 63 elements
0x014: 30f6980200000000000000 | irmovq dest, %rsi # dst array
0x01e: 30f7900000000000000000 | irmovq src, %rdi # src array
```

输出格式

- 对于每一个测试输入，输出一份同名的 `yml` 格式文件于 `output` 目录下，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 $\neq 0$ }，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，不用关心每次 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。

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

五、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**每组五分钟左右的汇报**，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

六、提交方式

内容要求

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（在讲清楚的前提下，越短越好）
- 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）

上传

```
git add -A
# 提交当前文件夹下的所有更改到暂存区
git commit -m "xxx(可以是你的提交注释)"
# 将暂存区的所有更改提交到本地仓库
git push
# 将本地仓库推送到远程
```

七、关于评分

- 实现Y86处理器（80%）
- 项目展示汇报（15%）
- 项目亮点和创新（5%）

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。**禁止抄袭代码**，鼓励自行学习相关知识丰富模拟器功能。

八、参考资料

- <http://csapp.cs.cmu.edu/3e/labs.html>
- 本文档编写时参考了22，23年的实验文档。