

用户手册

目录

一、产品介绍

二、安装指南

三、编译指南

（一）基本文件

（二）依赖包

1) NodeJS

2) CoffeeScript

四、使用说明

一、产品介绍

本产品为 y86 模拟器，能够接受指定格式的输入文件，并通过 Pipeline 机制执行该输入文件，同时将运行时状态以规定格式进行输出。

能够实现的功能有：

- 1) 基本功能：nop, halt, rrmovl, irmovl, rmmovl, mrmovl, addl, subl, andl, xorl, jmp, jle, jl, je, jne, jge, jg, call, ret, pushl, popl
- 2) Hazard 处理
 1. Data Hazard
 2. Control Hazard: Load/Use, Misprediction, Return, Combine
 3. Invalid
- 3) 其他功能：前进/回退；以不同速率执行文件

二、安装指南


本应用为纯前端 web 应用，无需发布无需安装，直接运行 dist/index.html 即可运行。已测试通过的浏览器有：Chrome 27, Firefox 16, IE 10

三、编译指南

本项目全部开源，如果有需要自定义的用户可以参考以下编译指南。

（一）依赖包：

1) NodeJS:

1. 打开 <https://github.com/joyent/node>，点击  下载文件。
2. 将下载文件进行解压缩。
3. 在 terminal 下定位到所解压缩的文件夹内，运行以下代码并耐心等待：

```
./configure
make
make install
```

4. 安装完成。

2) CoffeeScript:

1. 打开 <https://github.com/jashkenas/coffee-script>，点击  下载文件。

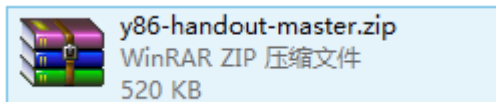
2. 将下载文件进行解压缩。
3. 在 terminal 下定位到所解压缩的文件夹内，运行以下代码并耐心等待：

```
sudo bin/cake install
```

4. 安装完成。

（二）基本文件：

1. 打开 <https://github.com/LazyChild/y86-handout>，点击  下载文件。



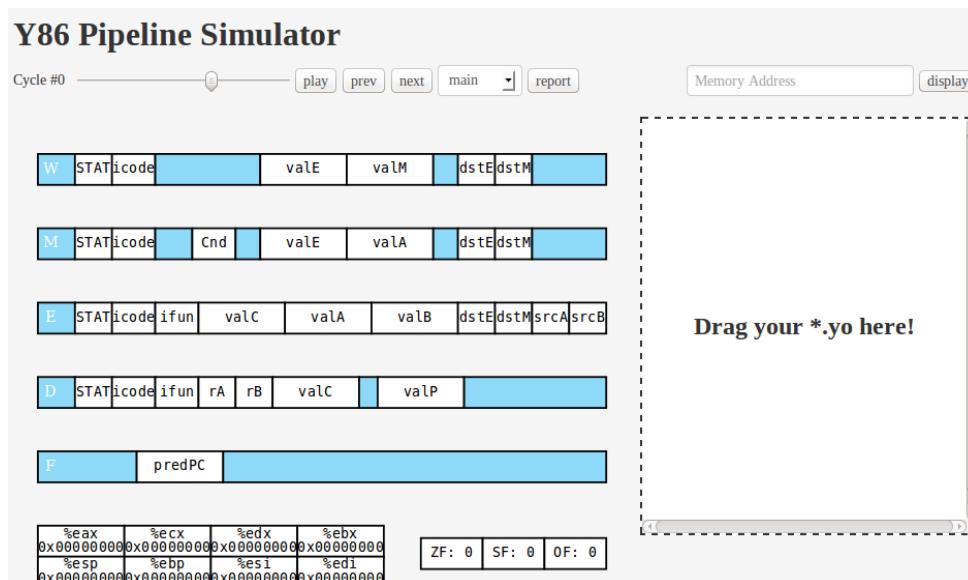
2. 将下载文件进行解压缩
3. 修改 src 下的 coffeescript 可以修改当前的代码。
4. 在 terminal 下定位到所解压缩的文件夹内，运行以下代码并耐心等待：

```
./run
```

5. 编译完成。

四、使用说明

1. 打开 dist/index.html，见到如下画面：



2. 将.yo 文件拖入 “Drag your *.yo here!” 框内。此时画面会变为下图：

Y86 Pipeline Simulator

Cycle #0 play prev next main report Memory Address display

W	BUB	icode	0x1	valE	valM	dstE	dstM	0xf	0xf
M	BUB	icode	0x1	Cnd	0x0	valE	valA	dstE	dstM
E	BUB	icode	ifun	0x0	valC	valA	valB	dstE	dstM
D	BUB	icode	ifun	0x0	rA	0xf	rB	0xf	valC
F	predPC 0x00000000								

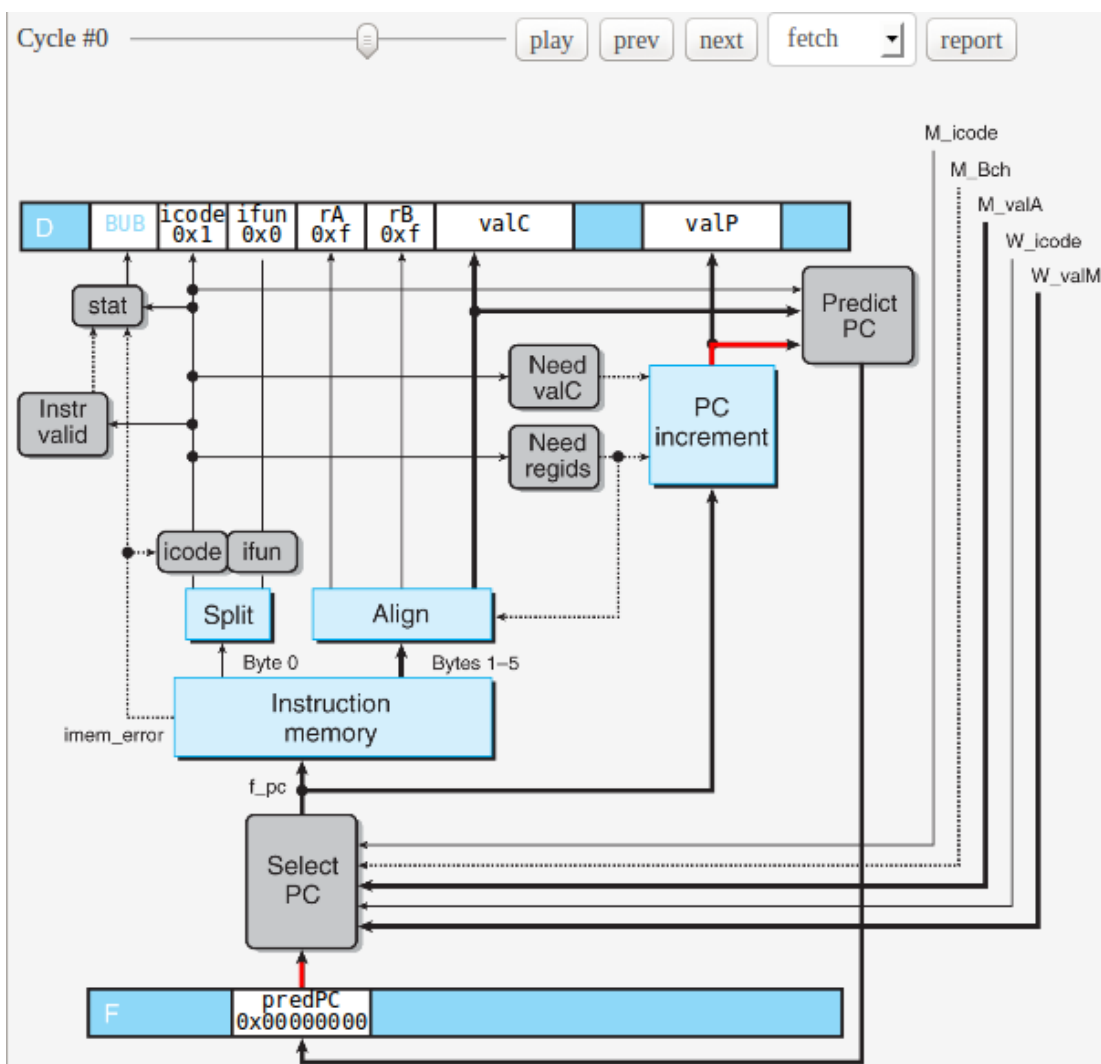
%eax	0x00000000	%ecx	0x00000000	%edx	0x00000000	%ebx	0x00000000
%esp	0x00000000	%ebp	0x00000000	%esi	0x00000000	%edi	0x00000000


ZF: 0 SF: 0 OF: 0

```


0x000 F init: irmovl Stack, %esp # Set
0x006      irmovl Stack, %ebp # Set
0x00c      call Main # Exe
0x011      halt # Ter
0x014 src: .long 0x00a
0x018 dest: .long 0x111
0x01c Main: pushl %ebp
0x01e      rrmovl %esp,%ebp
0x020      irmovl $1,%eax
0x026      pushl %eax # Pus
0x028      irmovl dest,%edx
0x02e      pushl %edx # Pus
0x030      irmovl src, %edx
0x036      pushl %edx # Pus
0x038      call copy_block # copy
0x03d      rrmovl %ebp,%esp
0x03f      popl %ebp
0x041      rat
  
```

3. 在此菜单 main 中选择不同的阶段会显示相应阶段的结构图及寄存器状态等：

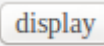


4. 点击  可执行下一周期，各阶段产生相应改变，同时右侧代码框中会标识当前的运行阶段：


```
0x000 D init: irmovl Stack, %esp    # Set
0x006 F      irmovl Stack, %ebp    # Set
```

5. 点击  可回滚上一周期，各阶段及代码处相应回复上一状态。

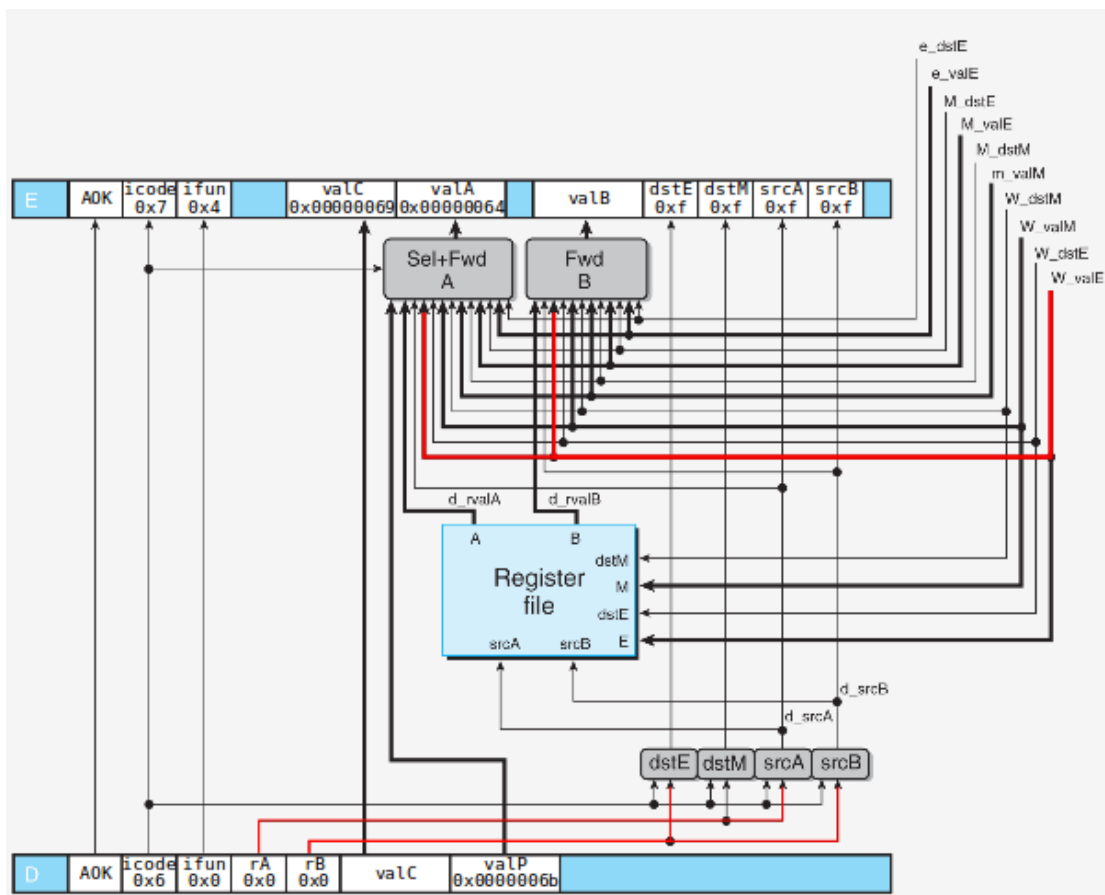
```
0x000 F init: irmovl Stack, %esp    # Set
```

6. 在 中输入所需要查看的 Memory 地址，点击  即可显示该地址中储存的值：

The content of address: 0x000000f0
is 0x00000018.

7. 点击  会自动顺序执行代码，点击  即可暂停。左侧会显示当前执行的周期，拖动进度条  可改变执行速度。

8. 在执行过程中，正在传递数据的线会显示为红色：



把鼠标移到部分元件上，还可以显示当前值：

`d_valA = 0x00000004`

9. 点击 可产生规定格式的输出文件：

Y86 Processor: Y86-CoffeeScript Full
153 bytes of code read

Cycle 0. cc = Z=false S=false O=false, STAT=AOK
F: predPC = 0x0
D: instr = nop, rA = ----, rB = ----, valC = 0x0, valP = 0x0, Stat = BUB
E: instr = nop, valC = 0x0, valA = 0x0, valB = 0x0
srcA = ----, srcB = ----, dstE = ----, dstM = ----, Stat = BUB
M: instr = nop, Cnd = false, valE = 0x0, valA = 0x0
dstE = ----, dstM = ----, Stat = BUB
W: instr = nop, valE = 0x0, valM = 0x0, dstE = ----, dstM = ----, Stat = BUB
Fetch: f_pc = 0x0, imem_instr = irmovl, f_instr = irmovl
Execute: ALU: + 0x0 0x0 --> 0x0

Cycle 1. cc = Z=false S=false O=false, STAT=AOK
F: predPC = 0x6
D: instr = irmovl, rA = ----, rB = %esp, valC = 0x100, valP = 0x6, Stat = AOK
E: instr = nop, valC = 0x0, valA = 0x0, valB = 0x0
srcA = ----, srcB = ----, dstE = ----, dstM = ----, Stat = BUB
M: instr = nop, Cnd = true, valE = 0x0, valA = 0x0
dstE = ----, dstM = ----, Stat = BUB
W: instr = nop, valE = 0x0, valM = 0x0, dstE = ----, dstM = ----, Stat = BUB