虚拟机设计文档

2016211397-翟泽鹏

**一：总述**

为了模拟64位虚拟机，自定义指令集，用函数的方式来实现CPU等功能，并且做出图形界面，不仅便于用户操作，还易于展示虚拟机执行过程。并且发布至Windows PC，Android，WEB三大平台，以期适应广大用户群体。

本工程最大特点就是界面实现力求华丽，精心设计UI、图片背景、音乐音效等元素，在设置中用户可以自主调节指令加入速度、指令运行速度、对比度、饱和度、亮度、声音大小等，并且点击设置里的帮助可直接下载本文档作为参考，注重用户交互体验，使新用户可快速上手。

**二：虚拟机开发简介**

开发者： 翟泽鹏

开发工具：unity 2017.3.1f1 + vs2015

开发环境：Windows 10

打开工程：在unity官网 <https://unity3d.com/cn>下载unity，新版unity也可打开旧版项目。

自己编写的C#源码在VM\_SourceCode/VM//Assets/Scripts目录下。

各平台下载网址：

Windows:

<https://github.com/zzp-seeker/vmPC/raw/master/VM_zzpseeker_SetUp.exe>

安卓:

<https://github.com/zzp-seeker/VM_android/raw/master/VM_zzpseeker.apk>

网页(pc端):

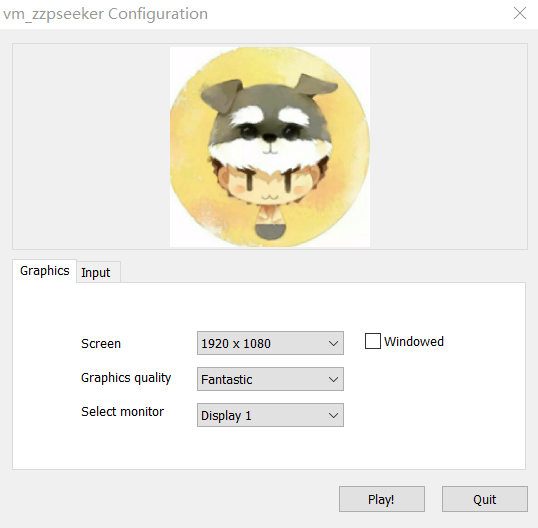
<https://zzp-seeker.github.io/ee/>

**三：虚拟机使用帮助**

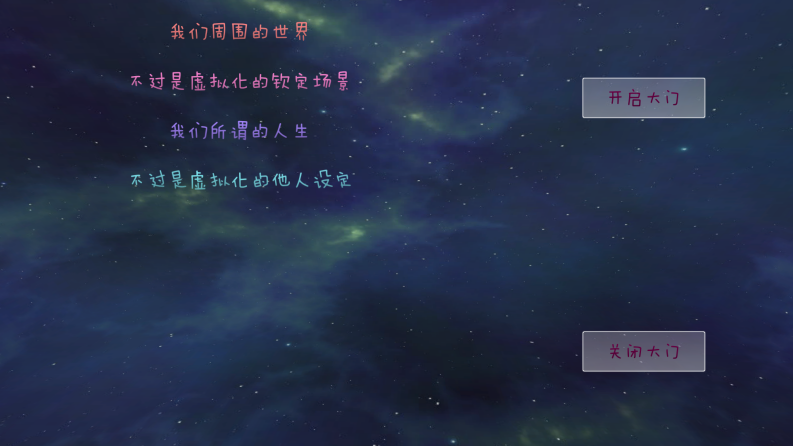
1.PC Windows

1)**安装完成后，打开VM\_zzpseeker.exe，进入如下界面，将Screen分辨率设为1920\*1080，**

**Graphics quality建议设为Fantastic，Select monitor设为Display 1。点击play!**



2)**打开后，出现之下界面，左边为动画式显示文本，鼠标左键点击屏幕任意处可下一处文字快速到来，点击开始大门进入虚拟机，点击关闭大门退出。**



3)**点击开始大门之后，进入如下界面，接下来介绍整个界面布局与各个按钮作用。**



整个界面设计华丽，背景是星空天空盒，随机旋转视角，带你领略星空的神秘与美丽。

4)**输入指令有两种方式，一是点击右上角的铅笔，出现如下界面。**



我们可以看见屏幕中间出现了一个面板，鼠标点击面板可以进行编辑，可手动输入，ctrl+c、ctrl+v、ctrl+a等常用文本编辑指令也皆可用。面板右边有几个按钮，前五个按钮是一些样例样例一是基本输入输出，样例二较为复杂的指令，样例三是斐波那契数列，样例四带有中断，样例五是错误样例，会输出异常情况，样例六是中断程序（带有中断字样）。

**编辑完成后，再次点击铅笔，即可按照设置中的指令加入速度加入指令。**

另外，在点击铅笔按钮后，若指令栏内没有指令，则编辑栏内默认显示样例1。若指令栏内有指令，则编辑栏内显示的即为指令栏内所对应的指令。如下图所示。



**第二种方式是通过屏幕下面的指令输入栏进行逐条输入。**

点击指令输入栏，即可手动输入指令。我们可以看见指令输入栏四角分别有个小按钮，以及指令输入栏右边有两个朝向不同的箭头。接下来说明它们功能。左上按钮：复制指令栏内容；左下按钮：在该指令栏最后粘贴内容；右上按钮：清空指令栏内容；右下按钮：将指令栏内容全复制并清空指令栏；朝右箭头：加入指令；朝左箭头：将指令栏最后一条内容拿出并加入指令栏。在指令栏输入时会有**输入字符限制**，**只能输入英文字母和数字**。

另外，鼠标悬浮于四角的按钮会显示帮助文本信息，而且可以结合以上两种方式进行指令输入。

5)**加入指令后，如下图所示，下图加入的是样例四，带有中断指令。**



下面我们说说如何浏览各个指令，首先整个指令面板可以通过最右的滑条或者鼠标放置滑条稍左或者小手那一竖条的位置进行鼠标滚轮滚动或者鼠标左键上下拖动可以拖动界面内容。另外鼠标位于某个指令栏中也可进行鼠标滚轮滚动或者鼠标左右拖动可拖动指令内容。在每个指令栏左面会有一个白色小框，点击后会让小手指向该指令。在整个指令栏左边的Introduce面板中会有小手指向的指令的功能简要介绍。

1. **接下来我们就可以运行指令了，点击屏幕正中间上方的运行按钮****，这时候指令就开始按照设置中的指令运行速度运行。**

此时虚拟机进入运行状态，在运行状态不能进行指令编辑等操作。在运行过程中，Memory面板会将内存中数据输出，Output面板会输出指令要求输出的内容，Register面板显示各个寄存器的内容，Introduce面板会显示该指令具体运行情况。另外，如不单击运行按钮，而是双击运行按钮，虚拟机直接进入暂停状态，单步执行。

**在运行过程中，可以随时点击暂停按钮****，虚拟机进入暂停状态。**

此时虚拟机进入暂停状态，在暂停状态下，指令并不会自动运行。我们可以点击上一步按钮，查看上一步指令运行情况，但是已经执行过的指令并不会再执行一遍，仅供调试时查看指令运行情况使用。我们也可以点击下一步按钮，若下一步指令并没有执行，则执行该指令，否则仅供查看该指令。

**在暂停过程中，也可以随时再次点击暂停按钮，虚拟机又进入运行状态。**

**在遇到STOP指令或异常情况时，我们可以点击停止按钮****，虚拟机处于就绪状态，这时候我们又可以编辑指令了。**

**总述：主要按钮功能**

1. ContactMe 

鼠标悬浮上面可以以动画方式显示一串文字

1. 上一步 暂停 运行 停止 下一步

**运行：**当虚拟机处于就绪状态时，点击运行即可按照设置中指令运行速度开始执行指令。

**暂停：**当虚拟机处于运行状态时，点击暂停即可进入暂停状态，再次点击该按钮即可进入运行状态。

**上一步：**当虚拟机处于暂停状态时，点击上一步可以查看上一步指令执行情况，但不会再次运行该指令。

**下一步：**当虚拟机处于暂停状态时，如果下一步指令未运行，点击下一步可以运行下一步指令，否则只是查看指令执行情况，但不会再次运行该指令。

**停止：**当虚拟机遇到STOP指令或者出现异常时，点击指令可停止虚拟机，使虚拟机重新进入就绪状态。

1. 设置 编辑

**其他设计**

**1.艾希英雄3d模型**



可以看见艾希下方分别由三个按钮，边上两个可以长按旋转艾希，中间按钮单击可以变换动作。

**2.虚拟机状态显示**

 分别有四个状态：就绪，运行中，暂停中，停止四个状态。

**3.Tips设计**

 在很多时候这里的tip会显示（如点击某个按钮，出现异常操作等），以此进一步增强用户交互体验。

**4.设置**



进入设置界面后，我们可以发现背景被模糊化，这是采用高斯模糊形成的，同时还有很多设置，点击help还可以获得本文档，十分人性化。

2.Android

该程序仅为主流屏幕分辨率1920\*1080设计，下载apk后，操作与pc相同，只要安卓系统较新，就不会有什么问题。以下为示意图。



3.网页(PC)

由于手机网页分辨率以及其他原因，目前不适合手机上打开网页。

若通过网址打开，刚开始会有加载页面，加载完毕后，建议点击主画面右下角的全屏按钮，然后操作与pc相同。

若想打开本地html文件，则需要浏览器支持webgl，则可以参考[http://www.hiwebgl.com/?p=628#WebGL-2](http://www.hiwebgl.com/?p=628" \l "WebGL-2) 该网页进行操作使浏览器支持webgl。



运行情况如上图所示，点击主画面右下角的全屏按钮，即可进入畅快体验。

**四：虚拟机设计描述**

64位CPU，1M内存，按字节寻址。

地址：00000到FFFFF，共2^20字节，

数据段：30000到CFFFF，共3 \* 2^18字节。

寄存器：共8个：AX，BX，CX，DX，EX，FX，GX，HX 。

CPU处理的数据皆为64位（二进制）无符号整数或字符串。

**1.操作数**

指令中操作数共3种形式：立即数，寄存器，内存。

1.1立即数：

一个16位的16进制常数，不会省略前导零，字母采用大写。如：02AC 0582 A598 C485 。

1.2寄存器：

通用寄存器：

AX，BX，CX，DX，EX，FX，GX，HX ，均为大写。

指令寄存器：IR

程序计数器：PC

堆栈指针寄存器：SP

状态字寄存器：SF （SF符号标志：当结果为负时为1，否则为0）

1.3内存：

有“立即数直接寻址”与“寄存器间接寻址”两种方式。

立即数直接寻址：T+立即数 如T02C0 。

寄存器间接间接寻址：T+寄存器 如TAX 。

存储模式：**小端存储**

整型数据的高字节保存在高地址中，数据的低字节保存在低地址中。该CPU处理的是64位数，占8个字节，故在内存中占相邻的8个地址。如读取某一地址的数据，则会读取到该地址以及其后7个相邻地址的数据，并将高地址作为高位，低地址作为低位，组成读取出来的64位数。如写入某一地址的数据，则会将64位数写入到8个相邻地址中。且高位写入高地址，低位写入低地址。

字符数据存储模式同整型数据，只不过字符串占用字节数不定，以\0标识结尾即可判别。

**2.指令设计**

共22种指令。

RUN：标识着程序的开始。如无特殊说明，内存和寄存器均已初始化为 0。

STOP：标识着程序的正常结束。

ECHO A：将操作数 *A*中的值输出。

ADD A B：将操作数 *A* 中的值与 *B* 中的值相加，结果存回 *A*。相加产生溢出时，直接将溢出部分丢弃即可（截断）—---无需向更高位进位，存回 *A* 的同样是一个16 位（二进制位）无符号整数。*A* 不能为立即数,*A*，*B*不能为字符串。

INC A ：将操作数 *A* 中的值加 1，结果存回 *A*。同样忽略溢出，*A* 不能是立即数。

SUB A ：将操作数 *A* 中的值减 1，结果存回 *A*。同样忽略溢出，*A* 不能是立即数。

AND A B：将操作数*A*与*B*进行与运算，结果存回*A*，*A*不能为立即数。

OR A B：将操作数*A*与*B*进行或运算，结果存回*A*，*A*不能为立即数。

XOR A B：将操作数*A*与*B*进行异或运算，结果存回*A*，*A*不能为立即数。

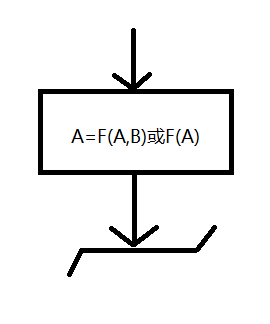
LSH A：将操作数*A*进行左移一位运算，结果存回*A*，*A*不能为立即数。

RSH A：将操作数*A*进行右移一位运算，结果存回*A*，*A*不能为立即数。

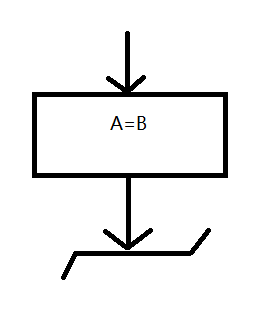
MUL A B：将操作数 *A* 中的值与 *B* 中的值相乘，结果存回 *A*。相乘产生溢出时，直接将溢出部分丢弃即可（截断）—---无需向更高位进位，存回 *A* 的同样是一个16 位（二进制位）无符号整数。*A* 不能为立即数。*A*，*B*不能为字符串。

DIV A B：将操作数 *A* 中的值与 *B* 中的值相除，结果存回 *A*。不能整除时，将小数点后丢弃，存回A的同样是一个16位无符号整数。*A*不能为立即数。*A*，*B*不能为字符串。

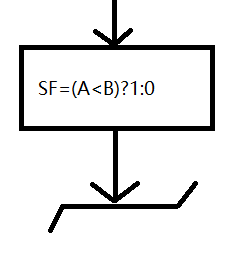
3-13 指令流程图 F代表某操作



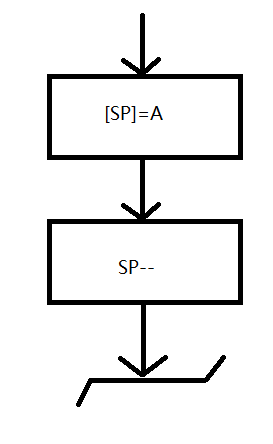
MOV A B ：将操作数 *B* 中的值写入 *A*，*A* 不能是立即数。



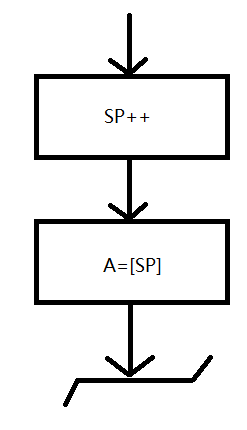
CMP A B：比较操作数 *A* 和 *B* 中的值的大小，结果存到状态字寄存器SF中。*A*，*B*必须同为整型数或字符串。



PUSH A：将操作数 *A* 中的值push入堆栈区。



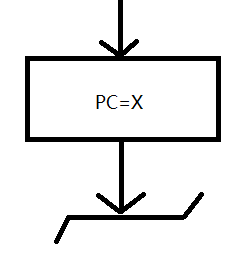
POP A：将栈顶数值pop给*A*，*A*不能是立即数。



跳转指令。

无条件跳转指令

JMP X 不妨假设操作数 *X* 中的值是 *i*，则该指令执行完后，将去执 行第 *i* 条指令。

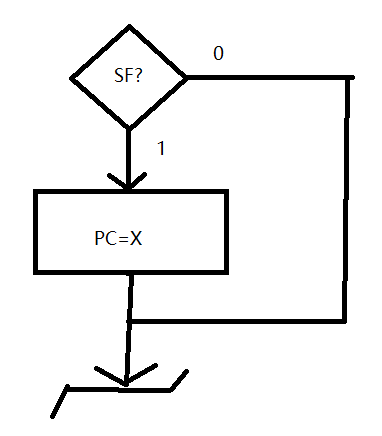


条件跳转指令

将状态字寄存器作为条件。

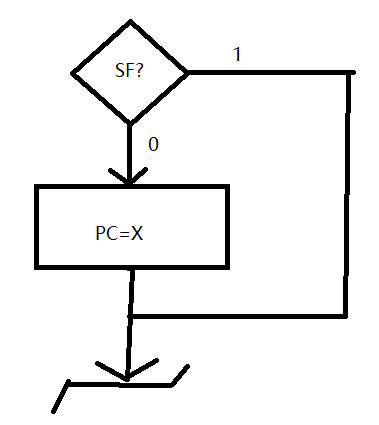
JSA X

当SF=1时跳转。



JSB X

当SF=0时跳转。



中断指令

INT：设置中断

RTI：从中断返回

特权指令

CRA：停止CPU工作

**指令总结**

操作指令中带有颜色的为操作数可以为字符串类型

|  |  |  |  |
| --- | --- | --- | --- |
| 指令类别 | 序号 | 操作码 | 功能 |
| 标识 | 1 | RUN | 标识程序开始 |
| 2 | STOP | 标识程序结束 |
| IO | 3 | ECHO A | 输出A |
| 4 | STO A | 存储A到IO |
| 5 | STD A | 读取IO并存入A |
| 操作 | 6 | ADD A B | A=A+B |
| 7 | INC A | A=A+1 |
| 8 | SUB A | A=A-1 |
| 9 | AND A B | A=A and B |
| 10 | OR A B | A=A or B |
| 11 | XOR A B | A=A xor N |
| 12 | LSH A | A<< |
| 13 | RSH A | A>> |
| 14 | MUL A B | A=A\*B |
| 15 | DIV A B | A=A/B |
| 16 | CMP A B | SF=(A<B)?1:0 |
| 17 | MOV A B | A=B |
| 跳转 | 18 | JMP X | PC=X |
| 19 | JSA X | if(SF) PC=X |
| 20 | JSB X | if(!SF) PC=X |
| 堆栈 | 21 | PUSH A | [SP]=A SP-- |
| 22 | POP A | SP++ A=[SP] |
| 中断 | 23 | INT | 设置中断 |
| 24 | RTI | 中断返回 |
| 特权 | 27 | CRA | 停止CPU运行 |

异常情形

存取非法ACCESS\_VIOLATION

跳转错误RUNTIME\_ERROR

特权指令ILLEGAL\_INSTRUCTION

不可识别指令 Instruction isn’t recognized

**五：代码设计与说明**

**六：感悟与体会**

自从老师布置下大作业，因为当时会一些unity，自己便开始着手用unity做大作业，一直做到现在才做到自己差不多满意的程度。在这期间，我碰到了很多很多的问题，也通过自主探索、上网搜索、询问别人、观看教学视频等方法解决了这些问题。已记不清有多少次熬到很晚，只为了解决一个很小却很棘手的问题；已记不清有多少次的激动，只是因为自己凭靠自己的努力实现了自己满意的效果。虽然过程充满坎坷，但我更多体会到的是一步一步实现图形化时自己的欣喜与激动。

也真的，非常非常感谢那些支持我的人，滴水之恩，当涌泉相报。我会记住，在这期间每一个鼓励过、赞美过我的人！！！