虚拟机设计文档

2016211397-翟泽鹏

一：总述

为了模拟64位虚拟机，可以自定义指令集，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。图形化界面中增加调节指令运行速度、实时停止程序运行等操作，可以实现更好的交互。并且发布至Windows PC，Android，IOS，WEB四大平台，以期适应广大用户群体。

二：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：

数字：

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 。

1. 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。

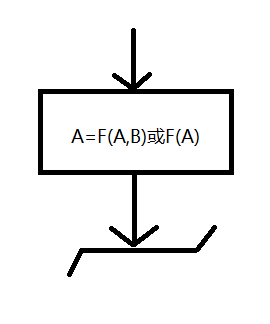
字符数据存储模式同整型数据，只不过字符串占用字节数不定，以\0标识结尾即可判别。

1. 指令设计

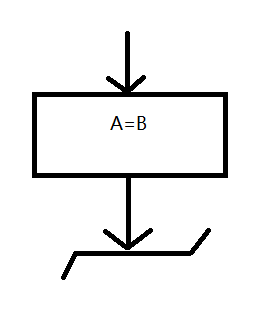
共22种指令。

1. 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。
2. STOP：标识着程序的正常结束。
3. ECHO A：将操作数 *A*中的值输出。
4. 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。
5. 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。
6. 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。
7. 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。
8. 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。
9. 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。
10. 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。
11. 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。
12. 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。
13. 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。

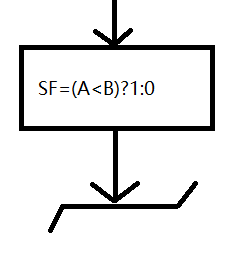
3-13 指令流程图 F代表某操作



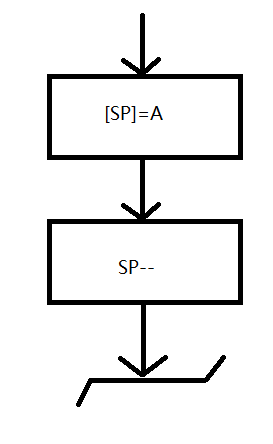
1. MOV A B ：将操作数 *B* 中的值写入 *A*，*A* 不能是立即数。



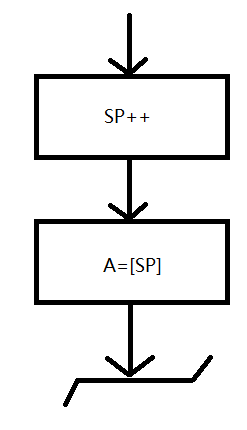
1. CMP A B：比较操作数 *A* 和 *B* 中的值的大小，结果存到状态字寄存器SF中。*A*，*B*必须同为整型数或字符串。



1. PUSH A：将操作数 *A* 中的值push入堆栈区。



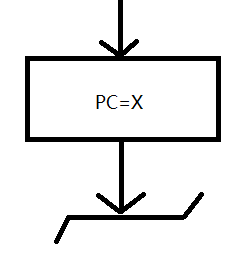
1. POP A：将栈顶数值pop给*A*，*A*不能是立即数。



1. 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。

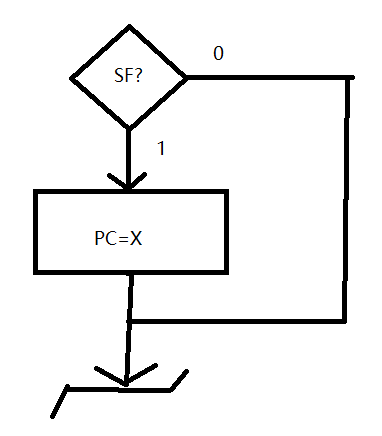


条件跳转指令

将状态字寄存器作为条件。

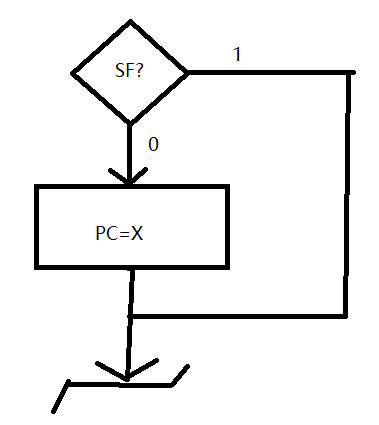
JSA X

当SF=1时跳转。



JSB X

当SF=0时跳转。



19.中断指令

INT：设置中断

RTI：从中断返回

20.程序控制类指令

POT：设置断点，之后程序会单步执行

NEXT：单步执行模式下，输入该指令会进行下一步指令

21.特权指令

CRA：停止CPU工作

1. IO指令

STO A：将A存入IO设备中

STD A：将IO设备中读出存入A，A不为立即数

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

1. 存取非法ACCESS\_VIOLATION
2. 跳转错误RUNTIME\_ERROR
3. 特权指令ILLEGAL\_INSTRUCTION
4. 不可识别指令 Instruction isn’t recognized