虚拟机操作说明书

**虚拟机安装说明：**

1. 开发者已经将所有.dll动态链接库文件，及原始可执行文件VM.exe一起打包放在文件夹“Virtual Machine”中。无需安装，可直接双击“VM.exe”运行。

2. 所有测试程序均保存在文件夹“测试程序”中。

**按键说明：**

『开机』：开启虚拟机，完成初始化。

『装载』：选择测试程序，并且加载到存储器的程序区。

『单拍』：单步执行1条指令。

『连续』：连续执行测试程序。

『滑块』：调整主频，控制连续执行的快慢。

『中断请求』：发送中断信号，请求中断。

『中断恢复』：发送中断恢复信号，从中断返回主程序。

『停机』：关闭虚拟机，清空界面。

『帮助』：显示帮助文档及测试教程。

『退出』：退出程序，关闭界面。

『输入』：从输入去输入数据。

『清空』：清空手动输入区的内容。

『执行』：将手动输入区的指令，加载到中断子程序，执行中断子程序。

**界面说明：**

1. 状态区：

（1） 状态：显示虚拟机当前所除状态。（0:未开机；1:开机未加载程序；2:程序加载完成；3:程序运行中；4:中断中；5:停机；）

（2） 异常：显示虚拟机运行过程中发生的错误。（0:程序运行正常；1:未加载程序时开始执行；2:指令包含非法操作数；3:寄存器超出规定数量；4:条件跳转时，未比较，但尝试跳转；5:在存储器中寻址时超出预设范围；6:弹栈时，堆栈里无数据；7:堆栈区溢出；8:陷入循环；9:中断中再次申请中断；10:未中断时，中断恢复；11:未中断时，手动输入指令；12:除数为0；）

（3）指令执行条数：记录从虚拟机运行开始，执行过的指令条数。

3. 程序显示区：

（1）测试程序：选择加载的测试程序，为主程序

（2） 中断子程序：手动输入的中断子程序。

4. 寄存器显示区：

（1）特殊寄存器：显示PC、IAR、IR、SP、PSW的值

（2） 通用寄存器：显示16个同用寄存器AX——PX的值。

5. 堆栈区：显示堆栈中的数据。

6. 输出区：输出到屏幕上。

7. 手动输入指令区：中断时，通过命令行手动输入中断子程序。

8. 日志区：显示并记录每一条指令的PC、操作过程和操作结果。

**测试程序说明：**

1. test.txt：测试程序包含指令集中所有指令。用于测试并验证指令正确性及虚拟机运行正确性。

2. error.txt：测试程序包含的指令均为错误指令。用于测试并验证虚拟机异常与检错的准确性。

3. Fibonacci\_Iteration.txt：测试程序为通过迭代的算法，求斐波那契数列前50项，并输出到输出区。用于验证跳转、循环等指令及虚拟机运行流程和逻辑的正确性。

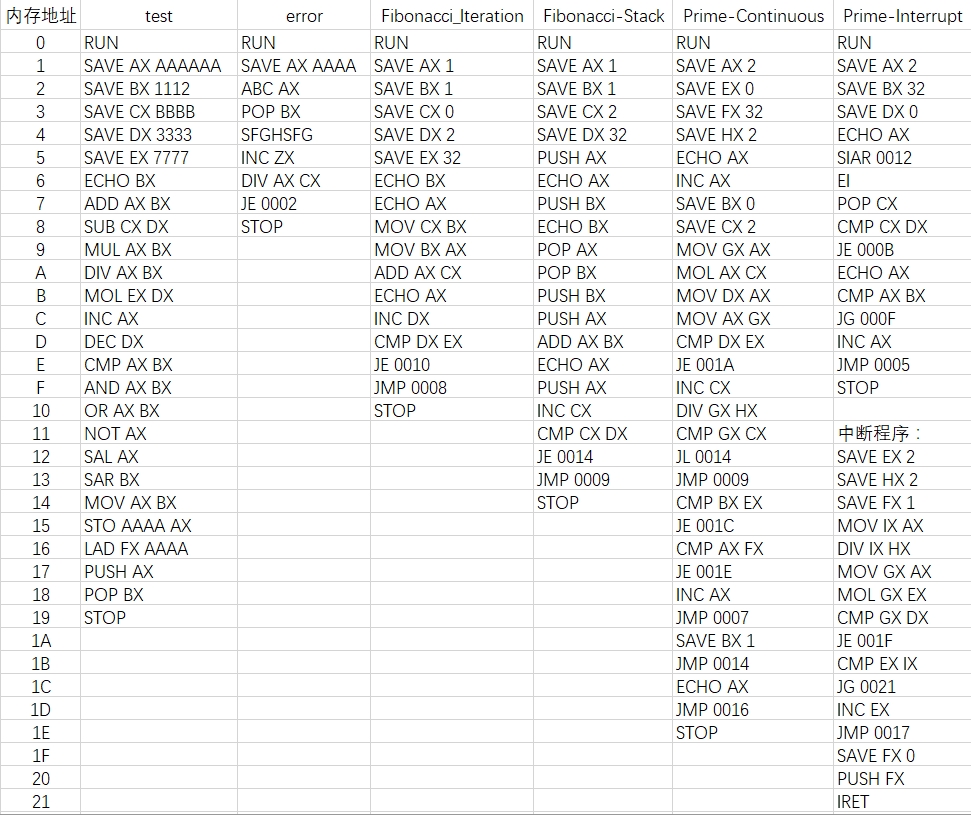
4. Fibonacci\_Stack.txt：测试程序为通过堆栈的操作，求斐波那契数列前50项，将斐波那契数列前50项依次压入堆栈区，并输出到输出区。用于验证入栈、出栈等指令及虚拟机维护堆栈区的操作逻辑的正确性。

5. Prime-Continuous.txt：测试程序为通过循环求50以内的质数，并输出到输出区。用于验证无条件跳转、条件跳转、比较、输出等指令及虚拟机执行程序中的双层循环操作逻辑的正确性。

6. Prime-Interrupt.txt：测试程序为通过中断子程序的方式求50以内的质数，并输出到输出区。为了测试特权指令，本测试程序加入了写中断子程序地址的指令，故请务必在开机之后，选择内核模式。用于验证主程序中断、跳转至中断子程序、中断恢复，函数间通过堆栈传递参数及特权指令逻辑的正确性。**由于选择内核模式，加入了特权指令。请在充分了解该测试程序的基础上，进行测试。否则如错误点击按钮、重复申请中断等操作，会引起不必要的错误与程序异常。**

7. 手动输入区程序：本虚拟机提供手动输入指令并执行的功能。单击中断申请后，主程序进入中断，可在手动输入区输入多条指令，指令格式参考“附件：指令集“。单击执行，将手动输入的程序加载进中断子程序，并执行。

具体测试程序如下：



**操作步骤：**

打开文件夹“Virtual Machine”，双击运行 “VM.exe”

0. 开始进行测试之前，建议单击**『帮助』**，查看虚拟机操作说明书及测试教程。

1. 单击**『开机』**，启动虚拟机。

2. 单击**『装载』**，选择文件夹“测试程序”。选择test.txt、error.txt、Fibonacci\_Iteration.txt、Fibonacci\_Stack.txt、Prime-Continuous.txt、Prime-Interrupt.txt六个测试样例之一，装载进虚拟机。

3. 若加载的测试程序为test.txt、error.txt、Fibonacci\_Iteration.txt、Fibonacci\_Stack.txt、Prime-Continuous.txt之一，则模式选择选中**『用户模式』**。若加载的程序为Prime-Interrupt.txt，则模式选择选中**『内核模式』**。

4. 若加载的测试程序为Fibonacci\_Iteration.txt、Fibonacci\_Stack.txt之一，该两个测试程序存在输入内容，在输入区**输入n**（n是在10和100之间的整数），单击**『输入』**。其余测试程序可忽略此步。

5. i：单击**『单拍』**，执行一条指令。

ii：单击**『连续』**，连续执行测试程序。左右调整滑块，可调整虚拟机主频，改变每一条指令执行速度。

6. test.txt、error.txt、Fibonacci\_Iteration.txt、Fibonacci\_Stack.txt、Prime-Continuous.txt测试过程中，可单击**『中断申请』**，从外部向虚拟机发送中断信号，中断测试程序。

Prime-Interrupt.txt测试过程中，由于已经存在中断子程序，并且在内核模式下测试，**请勿轻易点击『中断申请』**，以防造成不必要的错误。

7. 虚拟机处于中断中时，可在手动输入区输入多条指令，指令格式参考“附件：指令集“。

i：单击**『执行』**，将手动输入区的程序加载进中断子程序，并执行。

ii：单击**『清空』**，将手动输入区的内容清空。

8. 虚拟机处于中断中时，可单击**『中断恢复』**，恢复到测试主程序。

9. 测试程序执行完后，可单击**『停机』**，关闭虚拟机，清空所有数据。

10. 单击**『退出』**，退出虚拟机界面，关闭程序。

**测试结果：**

test.txt：

|  |  |  |
| --- | --- | --- |
| 内存地址 | 指令 | 执行结果 |
| 0 | RUN | START |
| 1 | SAVE AX AAAAAA | REGISTER AX : aaaaaa |
| 2 | SAVE BX 1112 | REGISTER BX : 1112 |
| 3 | SAVE CX BBBB | REGISTER CX : bbbb |
| 4 | SAVE DX 3333 | REGISTER DX : 3333 |
| 5 | SAVE EX 7777 | REGISTER EX : 7777 |
| 6 | ECHO BX | BX : 1112 |
| 7 | ADD AX BX | aaaaaa + 1112 = aabbbc |
| 8 | SUB CX DX | bbbb - 3333 = 8888 |
| 9 | MUL AX BX | aabbbc \* 1112 = b6278af38 |
| A | DIV AX BX | b6278af38 / 1112 = aabbbc |
| B | MOL EX DX | 7777 MOL 3333 = 1111 |
| C | INC AX | aabbbc + 1 = aabbbd |
| D | DEC DX | 3333 - 1 = 3332 |
| E | CMP AX BX | aabbbd > 1112 |
| F | AND AX BX | 101010101011101110111101 AND 1000100010010 = 1000100010000 |
| 10 | OR AX BX | 1000100010000 OR 1000100010010 = 1000100010010 |
| 11 | NOT AX | NOT 1000100010010 = 1111111111111111111111  111111111111111111111111111110111011101101 |
| 12 | SAL AX | 111111111111111111111111111111  1111111111111111111110111011101101 SAL = 1111111111111111111111111111111111  111111111111111101110111011010 |
| 13 | SAR BX | 1000100010010 SAR = 100010001001 |
| 14 | MOV AX BX | REGISTER AX : 889 |
| 15 | STO AAAA AX | REGISTER->MEMORY AAAA : 889 |
| 16 | LAD FX AAAA | MEMORY->REGISTER FX : 889 |
| 17 | PUSH AX | STACK c000 : 889 |
| 18 | POP BX | REGISTER BX : 889 |
| 19 | STOP | SHUT DOWN |

error.txt：

|  |  |  |
| --- | --- | --- |
| 内存地址 | 指令 | 执行结果 |
| 0 | RUN | START |
| 1 | SAVE AX AAAA | REGISTER AX : aaaaa |
| 2 | ABC AX | 操作数不合法 |
| 3 | POP BX | 弹栈时，堆栈里无数据 |
| 4 | SFGHSFG | 不合法指令 |
| 5 | INC ZX | 寄存器超出规定数量 |
| 6 | DIV AX CX | 除数为0 |
| 7 | JE 0002 | 条件跳转时，未比较，但尝试跳转 |
| 8 | STOP | SHUT DOWN |

Fibonacci\_Iteration.txt：

输出区输出：“1、1、2、3、5、8、13、21、34 ………” 斐波那契前n项。

Fibonacci\_Stack.txt：

输出区输出：“1、1、2、3、5、8、13、21、34 ………” 斐波那契前n项。

堆栈区0xC000-0xC031为“1、1、2、3、5、8、13、21、34 ………” 斐波那契前n项。

Prime-Continuous.txt、Prime-Interrupt.txt：

输出区输出：“2、3、5、7、11 ……… 43、47” 50以内的质数。

附件：指令集

• 立即数：一个16位的16进制常数，XXXXXXXXXXXXXXXX。不会省略前导零，字母使用大写，如 000002C002C002C0；  
• 寄存器：共16个通用寄存器。用AX、BX、CX、DX……PX表示，字母皆为大写。

• 内存地址：采用 ‘‘立即数直接寻址’’。通过4位16进制数 XXXX 表示，如02C0，表示内存地址02C0。

|  |  |  |
| --- | --- | --- |
| 开机 | 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。 |
| 停机 | STOP | 标识着程序的正常结束。 |
| 加法 | ADD AX BX | 将操作数 AX 中的值与 BX 中的值相加，结果存回 AX。相加产生溢出时，直接将溢出部分丢弃即可（截断）。 |
| 减法 | SUB AX BX | 将操作数 AX 中的值减去 BX 中的值，结果存回 AX。 |
| 乘法 | MUL AX BX | 将操作数 AX 中的值与 BX 中的值相乘，结果存回 AX。相乘产生溢出时，直接将溢出部分丢弃即可（截断） |
| 除法 | DIV AX BX | 将操作数 AX 中的值除以 BX 中的值，结果整数部分存回 AX。 |
| 模 | MOL AX BX | 将操作数 AX 中的值对 BX 中的值取模，结果存回 AX。 |
| 加1 | INC AX | 将操作数 AX 中的值加 1，结果存回 AX。同样忽略溢出。 |
| 减1 | DEC AX | 将操作数 AX 中的值减 1，结果存回 AX。 |
| 比较 | CMP AX BX | 比较操作数 AX 和 BX 中的值的大小，结果将存入 PSW，作为条件跳转指令的依据。 |
| 逻辑与 | AND AX BX | 将寄存器 AX 中的值与寄存器 BX 中的值按字节相与，结果存回寄存器 AX。 |
| 逻辑或 | OR AX BX | 将寄存器 AX 中的值与寄存器 BX 中的值按字节相或，结果存回寄存器 AX。 |
| 逻辑非 | NOT AX | 将寄存器 AX 中的值按字节取反，结果存回寄存器 AX。 |
| 左移 | SAL AX | 算术左移。把寄存器AX中数据的低位向高位移，空出的低位补0，再存回 AX。 |
| 右移 | SAR AX | 算术右移。把寄存器AX中数据的高位向低位移，空出的高位用最高位（符号位）填补，再存回 AX。 |
| 转移 | MOV AX BX | 将寄存器 BX 中的值写入寄存器 AX。 |
| 取数 | LAD AX XXXX | 将地址 XXXX 中的值写入寄存器 AX。 |
| 存数 | STO XXXX AX | 将寄存器 AX 中的值写入地址 XXXX。 |
| 置入存储器 | SET XXXX XXXX | 将立即数 XXXX 中的值置入内存地址 XXXX。 |
| 置入寄存器 | SAVE AX XXXX | 将立即数 XXXX 中的值置入寄存器 AX。 |
| 压栈 | PUSH AX | 将寄存器 AX中的值压入堆栈顶。 |
| 弹栈 | POP AX | 将栈顶数据弹出，存进寄存器 AX中。 |
| 无条件跳转 | JMP XXXX | 直接跳转。该指令执行完后，将去执行第 XXXX 条指令。 |
| 大于时跳转 | JG XXXX | a 大于 b 时跳转（PSW=1）。 |
| 小于时跳转 | JL XXXX | a 小于 b 时跳转（PSW=2）。 |
| 等于时跳转 | JE XXXX | a 等于 b 时跳转（PSW=3）。 |
| 中断允许 | EI | 申请中断。 |
| 中断返回 | IRET | 中断返回。 |
| 输出 | ECHO AX | 直接输出寄存器 AX 中数据到显示区。 |
| 取状态字 | LPSW AX | 读取状态字寄存器 PSW 中的数据，存入寄存器 AX 中。 |
| 存程序计数器 | SPC AX | 将寄存器 AX 中数据，存入程序计数器 PC 中。 |
| 读程序计数器 | LPC AX | 读取程序计数器 PC 中的数据，存入寄存器 AX 中。 |
| 读中断地址寄存器 | LIAR AX | 读取中断地址寄存器 IAR 中的数据，存入寄存器 AX 中。 |
| 存中断地址寄存器 | SIAR AX | 将寄存器 AX 中数据，存入中断地址寄存器 IAR 中。 |
| 空指令 | EMP | 空操作 |
| 清零 | CLR AX | 将寄存器 AX 中数据清零 |