



南开大学
Nankai University

南 开 大 学

计 算 机 学 院
计算机系统设计作业报告

PA 实验一报告

朱浩泽 1911530

年级：2019 级

专业：计算机科学与技术

指导教师：卢冶

2022 年 3 月 16 日

目录

一、 概述	1
(一) 实验目的	1
(二) 实验内容	1
二、 阶段一	1
(一) 实现正确的寄存器结构体	1
(二) 实现解析命令	2
(三) 实现单步执行	2
(四) 实现打印寄存器	3
(五) 实现内存扫描	4
三、 阶段二	5
(一) 表达式求值的计算	5
四、	5
(一) 第一节	5
(二) 第二节	5
(三) 第三节	7
五、 总结	7

一、 概述

(一) 实验目的

熟悉基础设施的各种工具和手段；熟悉寄存器间的存储关系；学习实现简易调试器补充指令；学习表达式求值，监视点的实现方法

(二) 实验内容

- 阶段一
 - 实现正确的寄存器结构体
 - 实现解析命令
 - 实现单步执行
 - 实现打印寄存器
 - 实现内存扫描
- 阶段二
 - 实现表达式求值
 - 实现词法分析
 - 实现递归求值
 - 实现调试中的表达式求值
- 阶段三
 - 实现断点
 - 实现监视点

二、 阶段一

(一) 实现正确的寄存器结构体

- 为了实现 32 位、16 位、8 位寄存器各 8 个和一个程序计数器 eip，我们利用匿名 Union 各个变量互斥并共享同一内存首地址这一特点，来实现这一结构体
- 修改 nemu/include/cpu/reg.h 中的代码如下

```
1 typedef struct {
2     union{
3         /* data */
4         union {
5             uint32_t _32;
6             uint16_t _16;
7             uint8_t _8[2];
8         } gpr[8];
9     } struct
10     {
11         rtlreg_t eax, ecx, edx, ebx, esp, ebp, esi, edi;
12     };
13 }
```

```

13     };
14     vaddr_t eip;
15 } CPU_state;

```

- 在上面的代码中，grp 中的的 _32 代表的是 eax 中的 32 位，_16 代表的是 ax 中的 16 位，_8[2] 分别代表的是 ah 中的 8 到 16 位和 al 中的 0 到 8 位

(二) 实现解析命令

- 在 nemu/src/monitor/debug/ui.c 中实现命令表，其代码如下

```

1 static struct {
2     char *name;
3     char *description;
4     int (*handler) (char *);
5 } cmd_table [] = {
6     { "help", "Display informations about all supported commands", cmd_help },
7     { "c", "Continue the execution of the program", cmd_c },
8     { "q", "Exit NEMU", cmd_q },
9
10    /* TODO: Add more commands */
11
12    { "si", "args:[N]; exectue [N] instructions step by step", cmd_si}, //让程序单步执
13    { "info", "args:r/w;print information about register or watch point ", cmd_info //行 N 条指令后暂停执行，当N没有给出时，缺省为1
14    }, //打印寄存器状态
15    { "x", "x [N] [EXPR];sacn the memory", cmd_x }, //内存扫描
16    { "p", "expr", cmd_p}, //表达式
17    { "w", "set the watchpoint", cmd_w}, //添加监视点
18    { "d", "delete the watchpoint", cmd_d} //删除监视点
19 };

```

- 其执行效果如下图所示

```

(nemu) help
help - Display informations about all supported commands
c - Continue the execution of the program
q - Exit NEMU
si - args:[N]; exectue [N] instructions step by step
info - args:r/w;print information about register or watch point
x - x [N] [EXPR];sacn the memory
p - expr
w - set the watchpoint
d - delete the watchpoint

```

(三) 实现单步执行

- 在 nemu/src/monitor/debug/ui.c 添加 cmd_si 函数，其代码如下

```

1 static int cmd_si(char *args) {
2     uint64_t N = 0;
3     if(args == NULL) {
4         N = 1;
5     }
6     else {
7         int temp = sscanf(args, "%lu", &N);

```

```

8   if(temp <= 0) {
9       printf("args error in cmd_si\n");
10      return 0;
11  }
12  }
13  cpu_exec(N);
14  return 0;
15  }

```

- 执行效果如下图所示，输入 si 和要执行的步骤 N，进行打印

```

lighthouse@VM-24-4-ubuntu:~/main/PA/ics2017/nemu$ make run
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default build-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 10:56:22, Mar 16 2022
For help, type "help"
(nemu) si 5
10000: b8 34 12 00 00      movl $0x1234,%eax
10005: b9 27 00 10 00      movl $0x100027,%ecx
1000a: 89 01                movl %eax,(%ecx)
1000c: 66 c7 41 04 01 00    movw $0x1,0x4(%ecx)
10012: bb 02 00 00 00      movl $0x2,%ebx
(nemu) █

```

(四) 实现打印寄存器

- 在 nemu/src/monitor/debug/ui.c 添加 cmd_info 函数，其代码如下，输入参数 r 对寄存器直接进行打印，输入参数 w 对监视点进行打印

```

1  static int cmd_info(char *args) {
2      char s;
3      if(args == NULL) {
4          printf("args error in cmd_info (miss args)\n");
5          return 0;
6      }
7      int temp = sscanf(args, "%c", &s);
8      if(temp <= 0) {
9          //解析失败
10         printf("args error in cmd_info\n");
11         return 0;
12     }
13     if(s == 'w') {
14         //打印监视点信息
15         print_wp();
16         return 0;
17     }
18     if(s == 'r') {
19         //打印寄存器
20         //32bit
21         for(int i = 0; i < 8; i++) {
22             printf("%s 0x%x\n", regsl[i], reg_l(i));
23         }
24         printf("eip 0x%x\n", cpu.eip);
25         //16bit
26         for(int i = 0; i < 8; i++) {
27             printf("%s 0x%x\n", regsw[i], reg_w(i));
28         }
29         //8bit

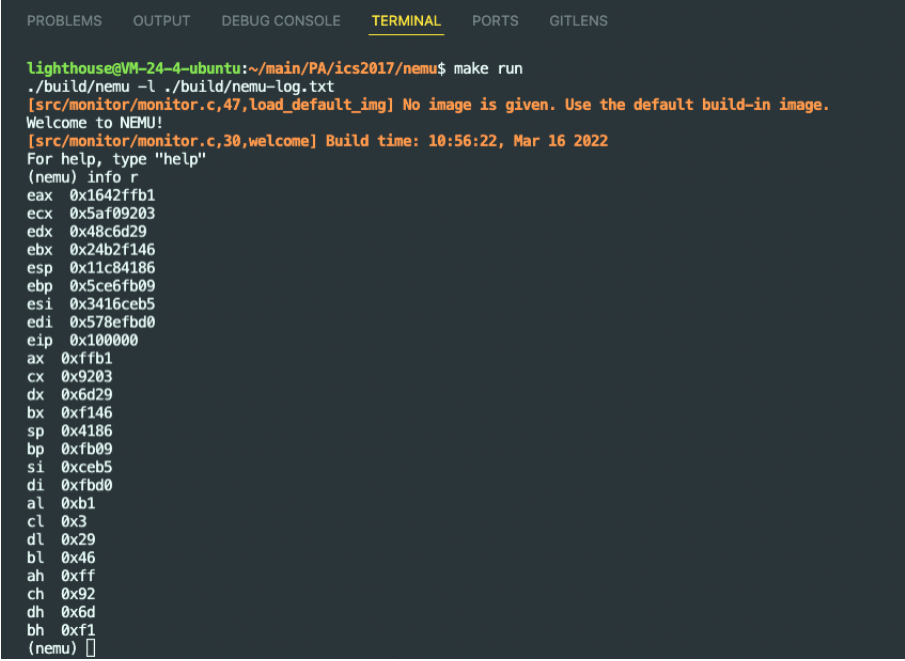
```

```

30     for(int i = 0; i < 8; i++)
31     {
32         printf("%s  0x%x\n", regsb[i], reg_b(i));
33     }
34     return 0;
35 }
36 //如果产生错误
37 printf("args error in cmd_info\n");
38 return 0;
39 }

```

- 执行效果如下图所示



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

lighthouse@VM-24-4-ubuntu:~/main/PA/ics2017/nemu$ make run
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default build-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 10:56:22, Mar 16 2022
For help, type "help"
(nemu) info r
eax  0x1642ffb1
ecx  0x5af09203
edx  0x48c6d29
ebx  0x24b2f146
esp  0x11c84186
ebp  0x5ce6fb09
esi  0x3416ceb5
edi  0x578efbd0
eip  0x100000
ax   0xffb1
cx   0x9203
dx   0x6d29
bx   0xf146
sp   0x4186
bp   0xfb09
si   0xceb5
di   0xfbd0
al   0xb1
cl   0x3
dl   0x29
bl   0x46
ah   0xff
ch   0x92
dh   0x6d
bh   0xf1
(nemu) 

```

(五) 实现内存扫描

- 在 nemu/src/monitor/debug/ui.c 添加 cmd_x 函数，其代码如下

```

1 static int cmd_x(char *args) {
2     int nLen = 0;
3     vaddr_t addr;
4     int temp = sscanf(args, "%d 0x%x", &nLen, &addr);
5     if(temp <= 0) {
6         //解析失败
7         printf("args error in cmd_si\n");
8         return 0;
9     }
10    printf("Memory:");
11    for(int i = 0; i < nLen; i++) {
12        if(i % 4 == 0) {
13            printf("\n0x%x:  0x%02x", addr + i, vaddr_read(addr + i, 1));
14        }
15        else {
16            printf("  0x%02x", vaddr_read(addr + i, 1));
17        }
18    }
19 }

```

```

18 }
19 printf("\n");
20 return 0;
21 }

```

- 该函数主要是调用了 nemu/src/memory/memory.c 中的 vaddr_read 函数进行内存扫描
- 执行效果如下图所示

```

lighthouse@VM-24-4-ubuntu:~/main/PA/ics2017/nemu$ make run
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default build-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 10:56:22, Mar 16 2022
For help, type "help"
(nemu) x 39 0x100000
Memory:
0x100000: 0xb8 0x34 0x12 0x00
0x100004: 0x00 0xb9 0x27 0x00
0x100008: 0x10 0x00 0x89 0x01
0x10000c: 0x66 0xc7 0x41 0x04
0x100010: 0x01 0x00 0xbb 0x02
0x100014: 0x00 0x00 0x00 0x66
0x100018: 0xc7 0x84 0x99 0x00
0x10001c: 0xe0 0xff 0xff 0x01
0x100020: 0x00 0xb8 0x00 0x00
0x100024: 0x00 0x00 0xd6
(nemu)

```

三、 阶段二

(一) 表达式求值的计算

在有了编译原理课程的基础后，我们可以知道，表达式求值的过程是先将整个句子切分为各个 token，然后对各个 token 按照赋予的优先级进行分割后求返回值，最终通过这种递归运算求出整个表达式的值。

在这个求值过程中，我们需要注意寄存器的值是可以当作操作数进行运算的，且对于括号的操作跟编译原理中直接利用 lex 工具

四、

(一) 第一节

如图1所示



图 1: Caption

表
带单元格表格

(二) 第二节

伪代码

N/n\Algo	naive-conv	naive-pool	omp-conv	omp-pool
64/2	0.0167	0.01255	0.04142	0.03799
64/4	0.03599	0.0394	0.0458	0.0421

表 1: 性能测试结果 (4 线程)(单位:ms)

<i>Cost</i>		To				
		<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
From	<i>B</i>	7	0	1	3	8
	<i>C</i>	8	1	0	2	7
	<i>D</i>	8	3	2	0	5

表 2: 结点 C 距离向量表 (无毒性逆转)

Algorithm 1 初始化 obj 文件信息——对应 MeshSimplify 类中 readfile 函数, Face 类 calMatrix 函数

Input: obj 文件, 顶点、边、面列表

Output: 是否读取成功

```

1: function calMatrix(Face)
2:    $normal \leftarrow e1 \times e2$ 
3:    $normal \leftarrow normal / normal.length$ 
4:    $temp[] \leftarrow normal.x, normal.y, normal.z, normal \cdot Face.v1$ 
5:    $Matrix[i][j] = temp[i] * temp[j]$ 
6:   return Matrix
7: end function
8: 根据 obj 的 v 和 f 区分点面信息, 读取并加入列表
9:  $scale \leftarrow$  记录点坐标中距离原点最远的分量, 以便后续 OpenGL 进行显示
10:  $ori \leftarrow$  记录中心点, 便于 OpenGL 显示在中心位置, 避免有的 obj 偏移原点较多
11: 根据三角面片信息, 计算一个面的三条边
12: 计算每个面的矩阵  $\leftarrow calMatrix$ 
13: 将每个面的矩阵加到各点, 由点维护
14: return True

```

代码

```

1 void ord()
2 {
3     double head,tail,freq,head1,tail1,timess=0; // timers
4     init(N);
5     QueryPerformanceFrequency((LARGE_INTEGER *)&freq );
6     QueryPerformanceCounter((LARGE_INTEGER *)&head);
7     for (int i=0; i<NN; i++)
8         for (int j=0; j<NN; j++)
9             col_sum[i] += (b[j][i]*a[j]);
10    QueryPerformanceCounter ((LARGE_INTEGER *)& tail) ;
11    cout << "\nordCol" <<(tail-head)*1000.0 / freq<< "ms" << endl;
12 }

```


逐列访问平凡算法

(三) 第三节

参考文献 [?] [?]

多行公式

$$a + b = a + b \quad (1)$$

$$\frac{a + b}{a - b} \quad (2)$$

行内公式: $\sum_{i=1}^N$ 超链接 [YouTube](#)

带标号枚举

1. 1

2. 2

不带标号枚举

• 1

• 2

切换字体大小

五、 总结