

数字逻辑与处理器基础——汇编实验报告

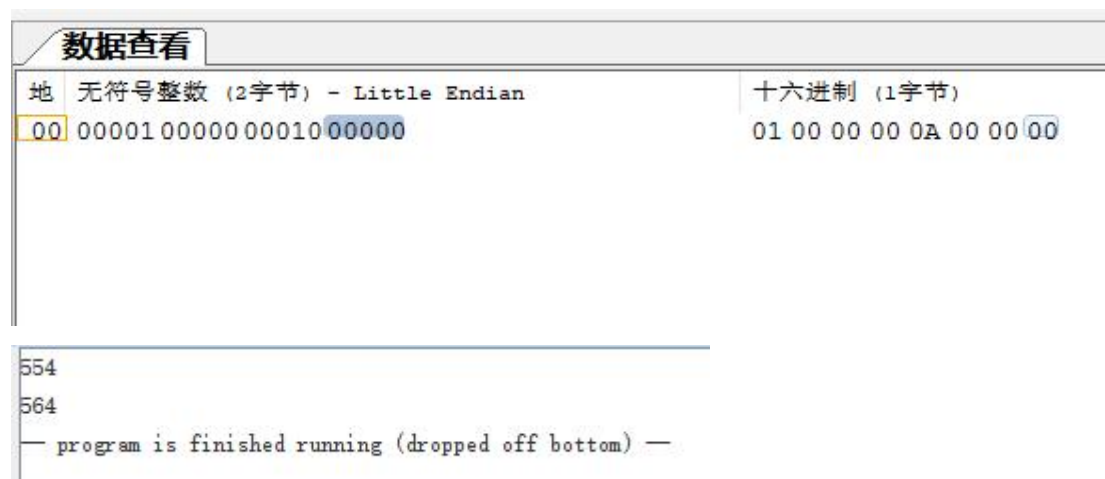
陈彦旭 2022010597

练习 1-1 系统调用

源代码见 exp1_1.asm

从文件 a.in 中读取两个整数，放在缓冲区 buffer 中，再将这两个数从 buffer 中写入文件 a.out
从键盘输入读取一个整数，将其加 10 后输出到屏幕。

测试样例：

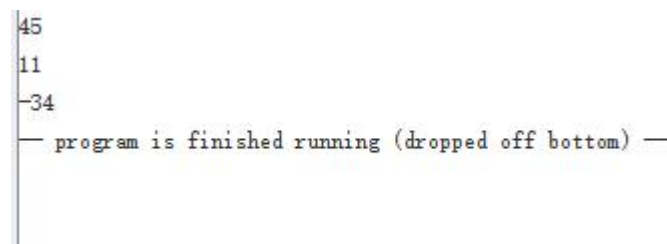


练习 1-2 循环，分支

源代码见 exp1_2.asm

从键盘输入两个整数 i 和 j，将 i 取相反数、j 取绝对值。从 i 开始循环 j 轮，每轮 i+=1，最后向屏幕输出 i，并保存在寄存器 \$v0。

测试样例：



练习 1-3 指针，数组

源代码见 exp1_3.asm

读入一个长度为 n 的整数数组，将数组逆序并把每个元素加 1 后输出

测试样例：

```

8
4
6
7
14
52
11
5
20
216125315-675
— program is finished running —

```

练习 1-4 函数调用

源代码见 exp1_4.asm

解决汉诺塔问题：运用递归函数

$$Hanoi(n) = \begin{cases} 1, & n = 1 \\ 2Hanoi(n-1) + 1, & n \geq 1 \end{cases}$$

从键盘屏幕读入整数 n ，输出 $Hanoi(n)$ 的值

测试样例：输入 11 得到结果 2047

```

11
2047
— program is finished running —

```

练习 2

插入排序：

源代码见 insert_sort.asm

.space 汇编命令为 buffer 开辟 4004 大小空间

读入文件 a.in，将所有数据装入缓冲区 buffer

buffer 地址装入 \$a0

\$a1 存储 N（通过 \$a0 取出 buffer[0]=N），作为 insertion_sort 的参数

\$t0 存储 compare_count（buffer[0]）

\$a0 自增 4，作为 insertion_sort 的参数

insertion_sort 函数由 3 部分组成：insertion_sort_ready, insertion_sort_loop, insertion_sort_end。调用 insertion_sort_ready 完成整个 insertion_sort 的调用，在 insertion_sort_loop 中调用 search 和 insert：

栈中保存 3 个寄存器：\$ra, \$a0, \$a1

\$t1 循环次数计数器 i
 \$a2 当前循环次数 i, 作为 search 的参数
 \$a3 由上面 search 的返回值得到 (插入位置), 作为 insert 的参数
 search 函数由 3 部分组成: search_ready, search_loop, search_end。调用 search_ready 完成对整个 search 的调用:
 栈中保存 3 个寄存器: \$ra, \$a0, \$a2
 \$t2: tmp=v[n] 的地址
 \$t3: tmp=v[n]
 \$t4: 循环次数计数器 i
 \$t5: v[i]的地址
 \$t6: v[i]
 insert 函数由 3 部分组成: insert_ready, insert_loop, insert_end。调用 insert_ready 完成对整个 insert 的调用:
 栈中保存 4 个寄存器: \$ra, \$a0, \$a2, \$a3
 \$t2: tmp=v[n] 的地址
 \$t3: tmp=v[n]
 \$t4: 循环次数计数器 i
 \$t5: v[i]的地址
 \$t6: v[i]
 最后写入文件, buffer 地址装入 \$s0, 系统调用将所有数写入 a.out

运行结果:

数据查看			
地址	无符号整数 (2字节) - Little Endian	十六进制 (1字节)	
00	0010300000011020000002559000000311500000	67	00 00 00 4E 04 00 00 FF 09 00 00 2B 0C 00 00
10	0474900000121000000014152000001461600000	8D	12 00 00 44 2F 00 00 48 37 00 00 18 39 00 00
20	1509000000154460000016658000001680800000	F2	3A 00 00 56 3C 00 00 12 41 00 00 A8 41 00 00
30	187730000036569000003906500000442500000	55	49 00 00 D9 8E 00 00 99 98 00 00 DA AC 00 00
40	4697900000500730000054452000005600900000	83	B7 00 00 99 C3 00 00 B4 D4 00 00 C9 DA 00 00
50	5629100000	E3	DB 00 00

文件名称: a.out 大小: 84 字节 地址: 00000000(Hex)/0(Dex) 选中部分: 1 字节

比较次数 0x67, 即 103 次, 排序后序列升序。

二分插入排序：

源代码见 `binary_insert_sort.asm`

`.space` 汇编命令为 `buffer` 开辟 4004 大小空间

先读入文件 `a.in`，将所有数据装入缓冲区 `buffer`

`buffer` 地址装入 `$a0`

`$a1` 存储 `N`（通过 `$a0` 取出 `buffer[0]=N`），作为 `binary_insertion_sort` 的参数

`$t0` 存储 `compare_count`（`buffer[0]`）

`$a0` 自增 4，作为 `binary_insertion_sort` 的参数

`binary_insertion_sort` 函数由 3 部分组成：`binary_insertion_sort_ready`, `binary_insertion_sort_loop`, `binary_insertion_sort_end`。调用 `binary_insertion_sort_ready` 完成整个 `binary_insertion_sort` 的调用，在 `binary_insertion_sort_loop` 中调用 `binary_search` 和 `insert`：

栈中保存 3 个寄存器：`$ra`, `$a0`, `$a1`

`$t1` 循环次数计数器 `i`

`$a1` 当前循环次数 `i`，作为 `binary_search` 的参数 `n`

`$a2` 为 0，作为 `binary_search` 的参数 `left`，调用完 `binary_search` 后作为 `insert` 参数 `n`

`$a3` 作为 `insert` 的参数 `right`，调用完 `binary_search` 后作为 `insert` 参数 `k`

`binary_search` 函数由 6 部分组成：`binary_search`, `binary_search_return`, `binary_search_continue_left`, `binary_search_continue_right`, `binary_search_recursion`, `binary_search_end`。调用 `binary_search` 完成对整个 `binary_search` 的调用，在 `binary_search_recursion` 中递归调用自身：

栈中保存 5 个寄存器：`$ra`, `$a0`, `$a1`, `$a2`, `$a3`

`$t2`: `mid=(left+right)/2`

`$t3`: `tmp=v[mid]`的地址

`$t4`: `tmp=v[mid]`

`$t5`: `v[n]`的地址

`$t6`: `v[n]`

`insert` 函数由 3 部分组成：`insert_ready`, `insert_loop`, `insert_end`。调用 `insert_ready` 完成对整个 `insert` 的调用：

栈中保存 4 个寄存器：`$ra`, `$a0`, `$a2`, `$a3`

`$t2`: `tmp=v[n]` 的地址

`$t3`: `tmp=v[n]`

`$t4`: 循环次数计数器 `i`

`$t5`: `v[i]` 的地址，最后作为 `v[k]` 的地址

`$t6`: `v[i]`，最后是 `v[k]`

最后写入文件，`buffer` 地址装入 `$s0`，系统调用将所有数写入 `a.out`

运行结果：

数据查看			
地址	无符号整数 (2字节) - Little Endian	十六进制 (1字节)	
00	0006200000011020000002559000000311500000	3E 00	00 00 4E 04 00 00 FF 09 00 00 2B 0C 00 00
10	04749000000121000000014152000001461600000	8D 12	00 00 44 2F 00 00 48 37 00 00 18 39 00 00
20	15090000000154460000016658000001680800000	F2 3A	00 00 56 3C 00 00 12 41 00 00 A8 41 00 00
30	1877300000036569000003906500000442500000	55 49	00 00 D9 8E 00 00 99 98 00 00 DA AC 00 00
40	46979000000500730000054452000005600900000	83 B7	00 00 99 C3 00 00 B4 D4 00 00 C9 DA 00 00
50	5629100000	E3 DB	00 00

比较次数 0x3E，即 62 次，排序后序列升序。

归并排序：

源代码见 merge_sort.asm

.space 汇编命令为 buffer 开辟 4004 大小空间

.word 汇编命令存放 compare_count

先读入文件 a.in，将所有数据装入缓冲区 buffer

\$s0: buffer 的地址

\$t0: N=buffer[0]

\$t8: compare_count 的地址

\$t9: 存储 compare_count 的值

\$s1: 主函数中 head 的地址

\$s2: 主函数中 pointer 的地址

\$t1: 循环次数计数器 idx

\$t2: buffer[idx] 的地址

\$t3: buffer[idx]

\$a0: msort 的参数 head

msort 由 4 部分组成: msort_ready, msort_return_head, msort_loop, msort_recursion。调用 msort_ready 完成对整个 msort 的调用，在 msort_recursion 中递归调用 msort 以及调用 merge。

栈中保存 4 个寄存器: \$ra, \$a0, \$s3, \$s4

\$s3: 保存 stride_2_pointer

\$s4: 保存 stride_1_pointer

\$t2 和 \$t3 均为临时装入链表节点值的寄存器

\$a1: 第一个递归的返回值，merge 的参数 l_head

\$a2: 第二个递归的返回值，merge 的参数 r_head

merge 由 8 部分组成：merge_ready, merge_loop_outer, merge_loop_inner1, merge_continue, merge_break, merge_loop_inner2,, merge_loop_insert, merge_end。调用 merge_ready 完成对整个 merge 的调用。

\$t1: merge 函数内 new 的数组的首地址

\$s5: 保存 p_left

\$s6: 保存 p_right

\$t3: 保存 p_right_temp (某一段内)

\$t4: 保存 temp_right_pointer_next (某一段内)

\$t2, \$t3, \$t4, \$t5, \$t6 中间会临时保存一些值

函数结尾 lw \$v0, 4(\$t1) 返回 (int *)rv

最后写入文件 a.out, 先通过 compare_count 的地址 \$t8 写入, 再通过 \$s2 的 pointer 指针步进逐个写入。其中 \$t7 临时保存文件描述符

运行结果:

数据查看	
地址 无符号整数 (2字节) - Little Endian	十六进制 (1字节)
00 00076 000000 01102 000000 02559 000000 03115 000000	4C 00 00 00 4E 04 00 00 FF 09 00 00 2B 0C 00 00
10 04749 000000 12100 000000 14152 000000 14616 000000	8D 12 00 00 44 2F 00 00 48 37 00 00 18 39 00 00
20 15090 000000 15446 000000 16658 000000 16808 000000	F2 3A 00 00 56 3C 00 00 12 41 00 00 A8 41 00 00
30 18773 000000 36569 000000 39065 000000 44250 000000	55 49 00 00 D9 8E 00 00 99 98 00 00 DA AC 00 00
40 46979 000000 50073 000000 54452 000000 56009 000000	83 B7 00 00 99 C3 00 00 B4 D4 00 00 C9 DA 00 00
50 56291 000000	E3 DB 00 00

文件名称: a.out 大小: 84 字节 地址: 00000000(Hex)/0(D) 选中部分: 1 字节

比较次数 0x4C, 即 76 次, 排序后序列升序。