数字逻辑与处理器基础

————汇编大作业

班级：无26

学号：2022010645

姓名：李卓航

日期：2024年5月4日

# 作业概述

在 MARS模拟器上，将指定的C/C++代码手动编译成 MIPS 汇编指令，然后汇编，运行，调试，完成能够解决给定问题的汇编程序设计。本次作业的目的在于理解汇编语言如何完成高级语言描述的算法，了解 MIPS处理器的硬件结构如何实现指令的需求，同时学会如何编写调试汇编程序。

# 二、基础练习部分

## （1）exp1\_1

**1.程序功能：**

(1)申请一个8 byte整数的内存空间。

(2)从“a.in”读取两个整数到buffer数组。

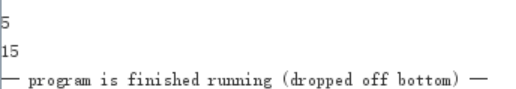
(3)向”a.out”写入buffer的两个整数。

(4)输入一个整数i,执行i = i +10，并打印结果。

**2.结果：**如图，成功从a.in读入了两个整数到buffer

****

如下图，输入5之后生成了15并打印出来



（2）exp1\_2

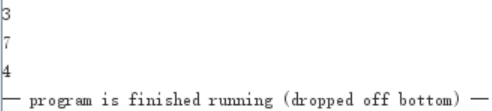
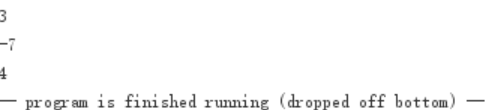
**1.程序功能：**

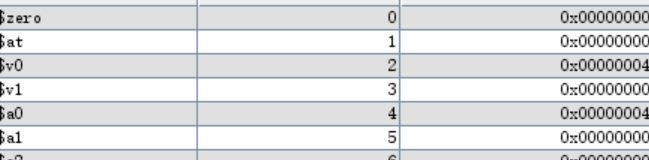
(1)将输入值i取相反数, 将输入值j取绝对值。（j为非0整数）

(2)从变量i开始，循环j轮，每轮i = i+1

即计算-i+|j|的值

**2.结果：**如图，输入3、7和3、-7进行测试，均得到了7-3=4的正确结果，储存在了v0中





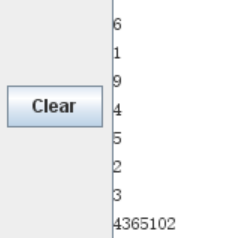
（3）exp1\_3

**1.程序功能：**

(1) 输入数组a的长度n（保证n为偶数）和任意n个整数

(2) 将数组a中的元素逐个加1，并且逆序存储在a中，最后打印数组a的值（中间不空）

**2.结果：**如图，输入6、1、9、4、5、2、3进行测试，得到了4365102的正确结果

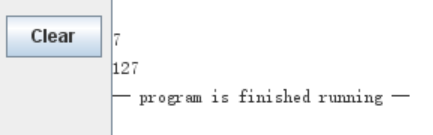


（4）exp1\_4

**1.程序功能：**

输入一个整数n，通过函数递归调用计算Hanoi（n）的值并打印输出

**2.结果：**如图，输入7进行测试，得到了127（）的正确结果

****

# 三、实战应用部分

（1）insert\_sort

**1.** **寄存器和变量之间的分配关系：**

$a0、$v0负责存储系统调用中的信息与$a1、$a2完成文件打开关闭与读写；

$s0存储buffer数组首地址，$s1存储N，$s2存储总比较次数，$s3存储search函数返回值，$s4存储search函数temp值，其在insert函数里作用类似；

$t0一般用于存储insertion\_sort函数for循环的循环变量i，$t2存储search函数返回值place在insertion\_sort函数中调用，$t4存储search函数循环变量i，$ t5存储search函数v[i]值，其在insert函数里作用类似，$t6负责在insert函数里存储v[i+1]地址用于sw操作。

其余还涉及到一些$sp栈操作或是小循环处$t的调用，因为细节太多不具体一一罗列，详情见代码中注释。

**2.** **过程调用操作：**

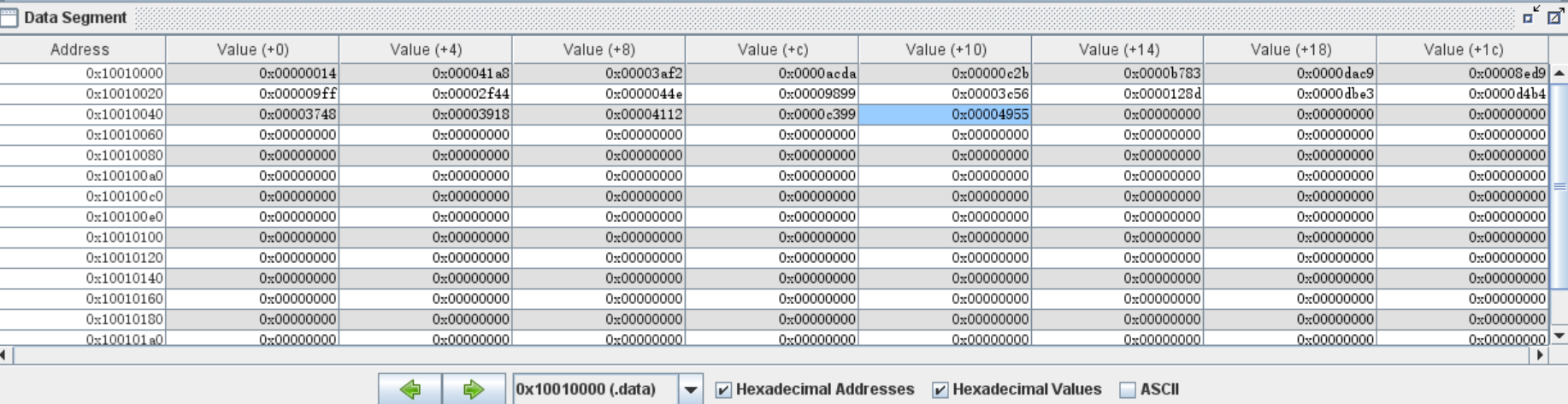
首先通过main函数进行文件读入存储到buffer数组中，接着加载N到$s1，以buffer[1]作为数组首地址传参到insert\_sort中，并在insert\_sort中直接修改buffer元素的值，最终将比较次数存入buffer[0]，写入到out文件里并关闭文件。

在insert\_sort函数中，进入for循环，先调用search函数找到i前面所有元素中第一个比v[i]小的位置，接着调用insert函数插入到该位置并将其后所有元素后移一位，完成排序；

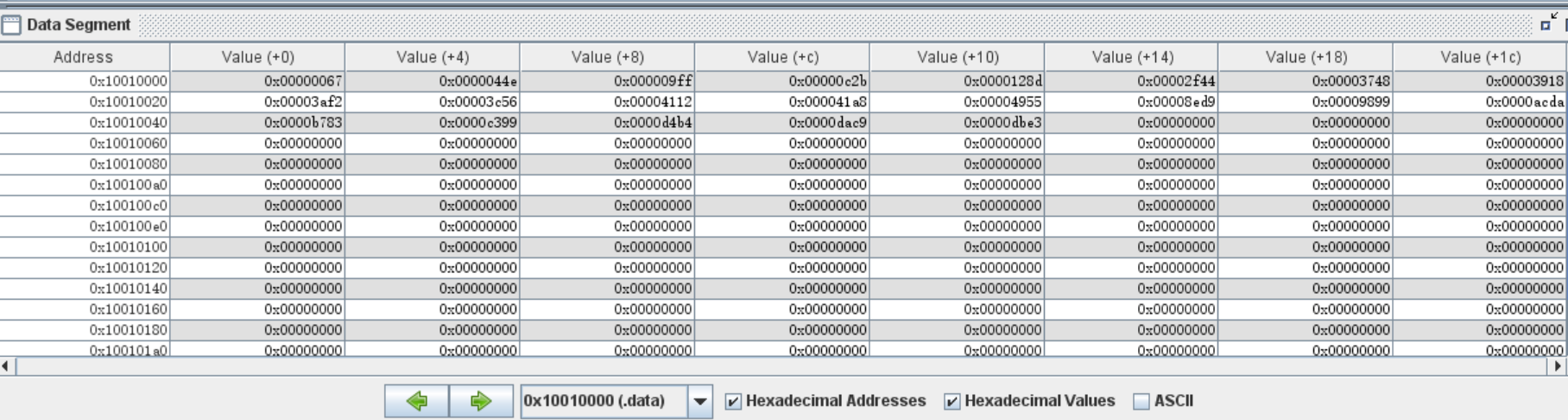
在search中使用for循环从后往前查找第一个比v[n]小的元素，查到则跳出循环，记录此时的位置，返回其后一位并返回insert\_sort函数作为place；在insert中先将v[i]值存储在temp中，接着依次该位置前面的值，将前面的值写入到后一位的地址中，直到place指定位置，完成元素后移，再将place位置的值加载为temp，完成i在place位置的插入，返回insert\_sort进行下一个i的操作。

**3.** **运行结果：**

**使用给定的a.in文件，加载其中N和N个待排序的数到buffer，结果如下：**

****

**排序完以后，将比较次数和排好后数的写入a.out文件，a.out用mars打开的结果如下：**

****

**第1位0x67=103，为排序次数，与对应cpp文件运行结果一致，而且发现后面的数均呈正序排列，可见完成了排序，且总体流程执行过程与cpp一致**

（2）binary\_ insert\_sort

**1.** **寄存器和变量之间的分配关系：**

$a0、$v0负责存储系统调用中的信息与$a1、$a2完成文件打开关闭与读写；

$s0存储buffer数组首地址，$s1存储N，$s2存储总比较次数，$s3存储binary\_search函数返回值，$s4存储binary\_search函数temp值，其在insert函数里作用类似，$s5、$s6、$s7分别存储binary\_search中left、right和mid指针；

$t0一般用于存储binary\_insertion\_sort函数for循环的循环变量i，$t2存储binary\_search函数返回值place在binary\_insertion\_sort函数中调用，$t4存储binary\_search和insert函数里形参n，$ t5存储binary\_search函数v[i]值，其在insert函数里作用类似，$t6负责在insert函数里存储v[i+1]地址用于sw操作。

其余还涉及到一些$sp栈操作或是小循环处$t的调用，因为细节太多不具体一一罗列，详情见代码中注释。

**2.** **过程调用操作：**

首先通过main函数进行文件读入存储到buffer数组中，接着加载N到$s1，以buffer[1]作为数组首地址传参到binary\_insert\_sort中，并在binary\_insert\_sort中直接修改buffer元素的值，最终将比较次数存入buffer[0]，写入到out文件里并关闭文件。

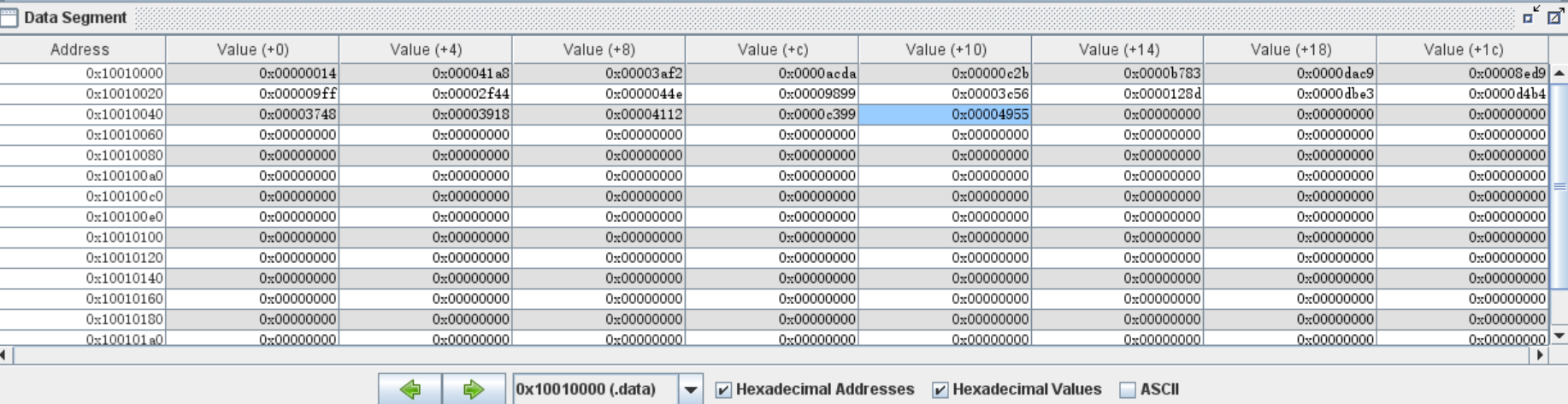
在binary\_insert\_sort函数中，进入for循环，先调用binary\_search函数找到i前面所有元素中第一个比v[i]小的位置，接着调用insert函数插入到该位置并将其后所有元素后移一位，完成排序；

在binary\_search函数中，先判断左右指针大小，如果左指针大则处于递归边界跳出循环进入binary\_search\_end返回左指针到$s3中，否则取左右指针中间值作为mid，比较v[mid]和当前位置v[i]大小，如果大于则二分查找左边，进入binary\_left，左指针不变，mid-1作为新的右指针再递归调用binary\_search，否则二分查找右边，进入binary\_right，右指针不变，mid+1作为新的左指针再递归调用binary\_search，最终找到指定位置返回binary\_insert\_sort。

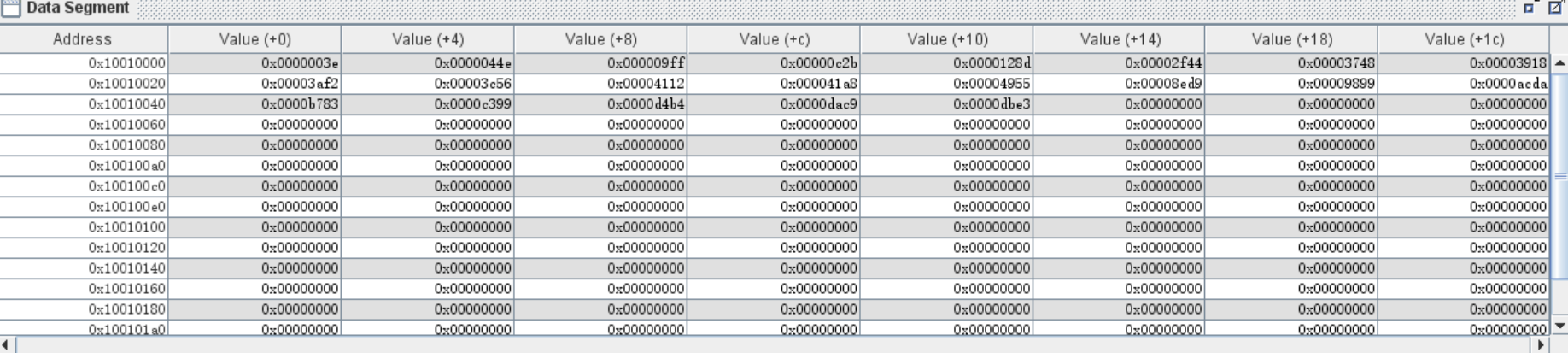
在insert中先将v[i]值存储在temp中，接着依次该位置前面的值，将前面的值写入到后一位的地址中，直到place指定位置，完成元素后移，再将place位置的值加载为temp，完成i在place位置的插入，返回binary\_insert\_sort进行下一个i的操作。

**3.** **运行结果：**

使用给定的a.in文件，加载其中N和N个待排序的数到buffer，结果如下：

****

排序完以后，将比较次数和排好后数的写入a.out文件，a.out用mars打开的结果如下：

****

第1位0x3e=62，为排序次数，与对应cpp文件运行结果一致，而且发现后面的数均呈正序排列，可见完成了排序，且总体流程执行过程与cpp一致

（3）merge\_sort

**1.** **寄存器和变量之间的分配关系：**

$a0、$v0负责存储系统调用中的信息与$a1、$a2完成文件打开关闭与读写；

$s0储存buffer[1]首地址，$s1存储N，$s2存储总比较次数, $s3存储pointer指针，$s4存储head头结点，$s5存储msort函数中head头结点；

$t0存储main函数创建链表for循环的循环变量idx，$t1存储new后分配的结点地址，$t3存储创建链表中buffer[idx]值，$t6存储排序后的链表的head，$t9存储最后写入文件过程中pointer[1]；

在msort函数里，$t2存储快指针stride\_2\_pointer，$t1存储慢指针stride\_1\_pointer，$t4存储stride\_1\_pointer[1]，$t3存储stride\_2\_pointer[1]，$t6储存msort返回结点，$s6存放l\_head,$s7存放r\_head用于后续merge函数；

在merge函数里，$s6存放虚拟头结点head，$t1存放左链表指针p\_left，$t2存放右链表指针p\_right，循环过程中$t7存储p\_right\_temp，$t8存储p\_left[1]，$t9存储p\_right[1]，循环结束后$t7储存最终返回结点rv用于msort调用；

其余还涉及到一些$sp栈操作或是小循环处$t的调用，因为细节太多不具体一一罗列，详情见代码中注释。

**2.** **过程调用操作：**

main同其他两个排序类似，加载完buffer后进入loop创建链表，如果计数器大于 N，则结束循环进入loop\_exit,先加载s5作为head跳转到msort函数中对链表进行排序，最终排完序后加载排序后链表头结点通过pointer将比较次数和排序后的值写入到out文件并关闭；

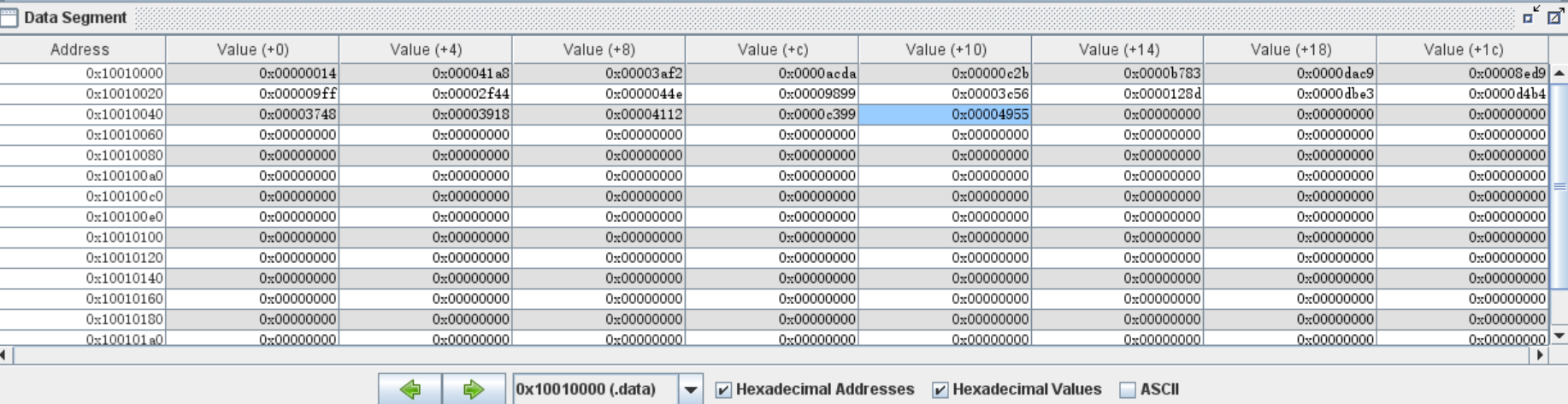
在msort函数里若head[1]为空则不在分割，进入msort\_exit返回head，否则进入到循环msort\_loop里移动快慢指针找到中点，若快指针到终点，则退出进入msort\_loop\_exit跳转回msort里面，将慢指针后一个位置置为新头结点，和原来的头结点分别再调用msort函数进行递归，划分，子链表排完序后返回头结点至父过程，最终将通过merge来将两个子链表按照正序合并；

在merge函数里先构建虚拟头结点，将左链表头结点作为新链表头结点，进入merge\_outer\_loop外循环，其中先进入内循环merge\_inner\_loop寻找左链中的插入位置，然后判断p\_left[1]是否为空，若为空则进入merge\_outer\_loop\_exit1跳出循环，将后续右链表全部接入该位置后方，否则进入内循环merge\_inner\_loop1寻找右链插入位置，最终合并左右链表，判断p\_right[1]是否为空，若为空则合并完成进入merge\_outer\_loop\_exit跳出循环，返回到父过程msort中作为新的子链表再进行合并。

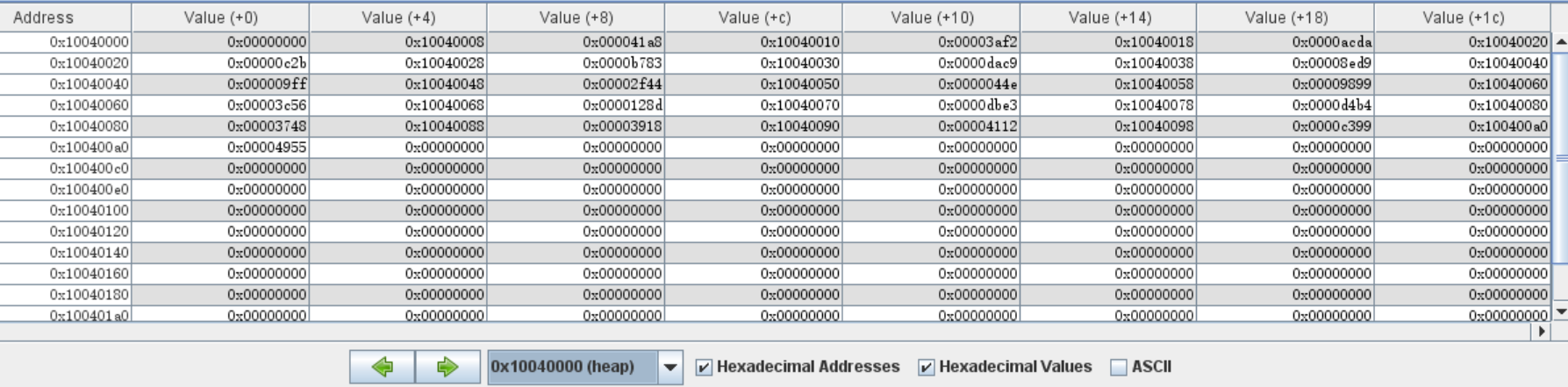
在merge\_inner\_loop内循环中，先判断p\_left[1]是否为空，为空则进入merge\_inner\_loop\_exit跳转回外循环merge\_outer\_loop；否则判断两个指针此时的对应值，若第一个对应值大于第二个则进入merge\_inner\_loop\_exit跳转回外循环merge\_outer\_loop，否则将第一个指针移动到下一个位置，往复循环。

**3.** **运行结果：**

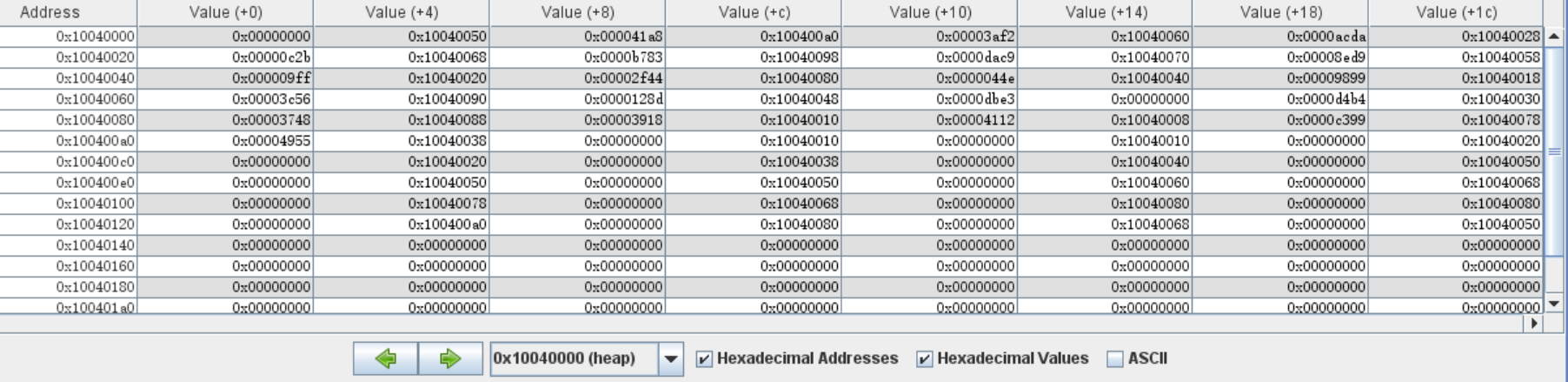
使用给定的a.in文件，加载其中N和N个待排序的数到buffer，结果如下：

****

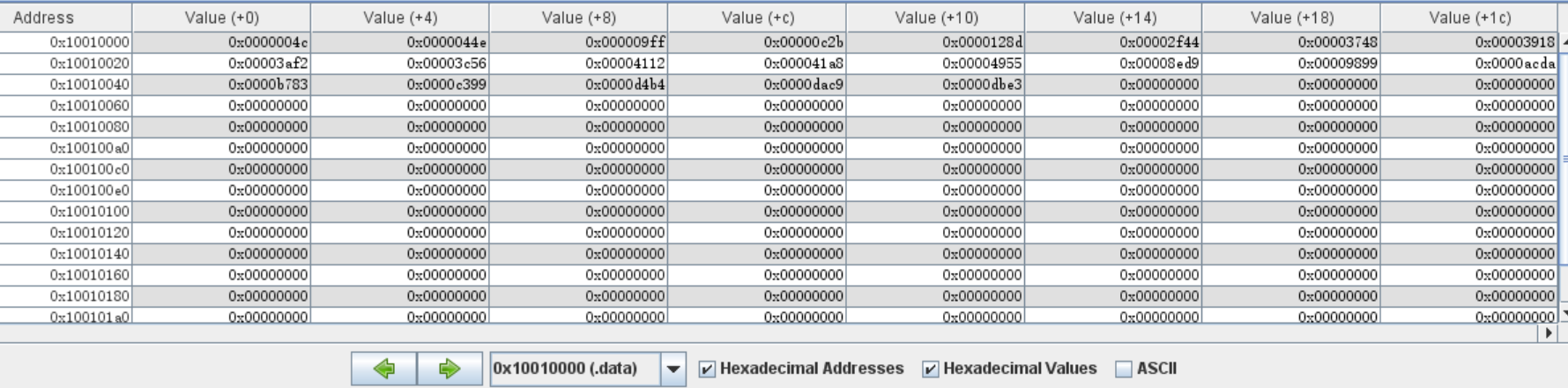
归并排序先构造好链表，如下图所示，在链表中每个结点均占据8byte，2word的空间，除去head以外第一个字储存着待排序的值，第二个字储存着下一个结点的首地址（即next指针）：

****

排完序后的链表如下图所示，可知此时链状结构已经按照值的大小呈正序排好（0x100400a8以后的为调用过程新开辟的虚拟头结点，不算在链表里）：

****

排序完以后，将比较次数和排好后数的写入a.out文件，a.out用mars打开的结果如下：

****

第1位0x4c=76，为排序次数，与对应cpp文件运行结果一致，而且发现后面的数均呈正序排列，可见完成了排序，且总体流程执行过程与cpp一致