数字逻辑与处理器作业
——汇编程序设计

2021 年 4 月 26 日

一、作业内容

本次作业要求同学们在 MARS 模拟器上，将指定的 C++代码翻译成 MIPS 汇编指令，然后编译，运行，调试，并通过测试。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。

1. 基础练习

练习 1-1：循环，分支

用 MIPS 语言实现 exp1_1_loop.cpp 中的功能并提交汇编代码，尽量在代码中添加注释。exp1_1_loop.cpp 代码内容主要包括：

- 将输入值取绝对值，存在变量 i，j 中
- 如果$j > i$，交换 i，j
- 求和 sum=0+1+2+……+j

练习 1-2：系统调用

练习使用 MARS 模拟器中的系统调用 syscall，使用 syscall 可以完成包括文件读写，命令行读写（标准输入输出），申请内存等辅助功能。系统调用基本的使用方法是

- 向$a*寄存中写入需要的参数（如果有）
- 向$v0 寄存器中写入需要调用的 syscall 的编号
- 使用”syscall”指令进行调用

- 从$v0 中读取调用的返回值（如果有）

更多具体的使用方法可以参照 MARS 模拟器的 Help 中的相关内容。exp1_2_sys_call.cpp 代码内容主要包括：

- 申请一个 8byte 整数的内存空间。
- 从”a.in”读取两个整数。
- 从键盘输入一个整数。
- 对上面 三个整数求最大值。
- 向屏幕打印最大值结果。
- 向”a.out”写入最大值结果。

输入输出文件格式： 输入文件名为“a.in”，输出文件名为“a.out”，输入输出文件均使用二进制格式。文件中提供了“a.in”作为测试样例，包含 1 和 10 两个整数。对于打开文件的 mips 指令，**只读对应$flag = 0$，只写对应$flag = 1$，可读可写对应$flag = 2$**

练习 1-3：数组，指针

用 MIPS 汇编指令实现 exp1_3_array.cpp 的功能并提交汇编代码，尽量在代码中添加注释。exp1_3_array.cpp 代码内容主要包括：

- 输入数组 a 的长度 n
- 任意输入 n 个整数
- 将数组 a 逆序，并且仍然存储在 a 中
- 打印数组 a 的值

提示： MIPS 中系统调用 9 和 C 语言中的 new 作用类似。使用该指令开辟 n 个整数的空间，传入参数为 n*4，返回值为空间首地址

练习 1-4：函数调用

本节提供了计算斐波那契数列的 C 语言代码，请补充完成 MIPS 代码，逐步完成函数调用的编译，使得其可以完成计算$Fib(n)$的任务。（在实验报告中完成即可，不需要提交相应汇编程序）

int Fib(int n){	Fib:#将参数放入$a0
	（保护现场）
s0 = n;	addi $s0 $a0 0
if(s0<3)	slti $t0 $s0 3
return 1;	beqz $t0 Next
	addi $v0 $0 1 # 返回1
	（恢复现场）
	jr $ra
else{	Next:
s1 = 0	addi $s1 $0 0
s1 += Fib(s0-1)	（调用Fib(n-1)）
	add $s1 $v0 $s1
s1 += Fib(s0-2)	（调用Fib(n-2)）
	add $s1 $v0 $s1
return s1;}	addi $v0 $s1 0
}	（恢复现场）
	jr $ra

提示：Fib(4)共计五次函数调用

$$Main() -> Fib(4),$$

$$Fib(4) -> Fib(3) + Fib(2),$$

$$Fib(3) -> Fib(2) + Fib(1)$$

2. 综合练习

练习 2：背包问题

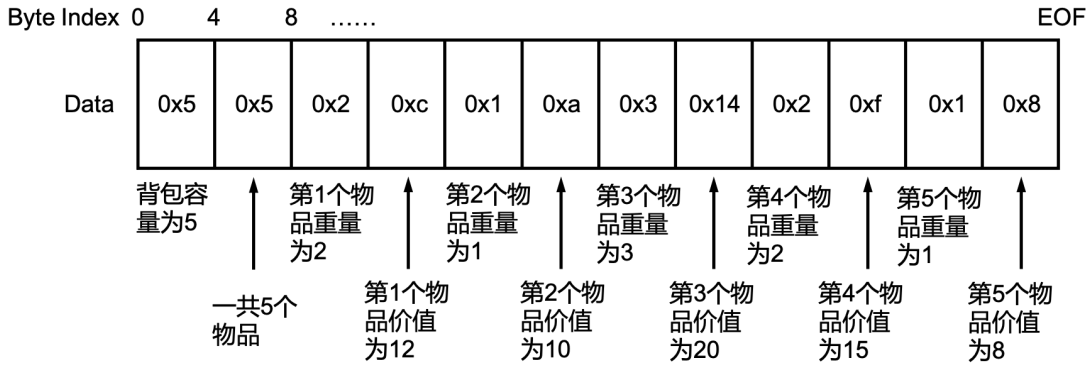
给定n个重量为$w1, w2, \cdots, wn$，价值为$v1, v2, \cdots, vn$的物品和一个承重量为W的背包，如果选择一些物品放到背包中，求使得背包中物品价值最大的方案。

例：背包总重量 W=5

序号	1	2	3	4	5
重量	2	1	3	2	1
价值	12	10	20	15	8

此时最优方案为：物品 2+物品 3+物品 5，重量 5，价值 38

输入文件格式： 输入文件为二进制文件。文件中第一个 4Bytes 表示背包的容量，第二个 4Bytes 表示总共的物品数量，之后的数据物品重量和价值交错排列，如下图：



本题的 C 语言代码分为三个版本：

（1）动态规划-自底向上，熟悉基本操作，练习文件读取写入。

（2）遍历搜索，练习位的相关操作。

（3）动态规划-自顶向下，练习递归函数调用，入栈出栈等操作。

用 MIPS32 汇编指令将三个版本的 C 语言代码**均转化**为汇编语言执行，调试代码并获得正确的结果。**测试样例被限制在最大背包总重量小于 63，物品数量小于 31。要求汇编程序结束时，计算结果储存在寄存器$v0 中。**

3. 作业要求

作业用一个压缩包提交，压缩包名称：”学号_姓名.7z”。推荐用 7z 格式，其他常见压缩格式也可以。

压缩包打开后需要包含:

一个“实验报告.pdf”文件，

一个“exp_1_1.asm”,一个“exp_1_2.asm”,一个“exp_1_3.asm”

一个“exp_2_1.asm”,一个“exp_2_2.asm”,一个“exp_2_3.asm”

注意所有的 MIPS 代码需要和 C 语言代码对应，不可使用其他 C 程序。实验 2 要求的输出格式：最终程序运行结束时结果储存在寄存器$v0 中。没有按要求输出将会酌情扣分。

二、附录

C++代码

exp1_1_loop.cpp

```
#include "stdio.h"
int main()
{
int i,j,temp,sum=0;
scanf("%d",&i);
scanf("%d",&j);
if (i<0){i=-i;}
if (j<0){j=-j;}
if (j>i){
temp = i;
i = j;
j = temp;
}
for(temp=0;temp<=j;++temp)
{
sum += temp;
}
```

```
printf("%d",sum);
return 0;
}
```

exp1_2_sys_call.cpp

```
#include "stdio.h"
int main()
{
    FILE * infile ,*outfile;
    int i,max_num=0,id;
    int* buffer;
    buffer = new int[2];
    infile = fopen("a.in","rb");
    fread(buffer, 4, 2, infile);
    fclose(infile);
    scanf("%d",&i);
    max_num = i;
    for(id=0;id<2;++id)
    {
     if(max_num < buffer[id])
     {max_num = buffer[id];}
    }
    buffer[0] = max_num;
    printf("%d",buffer[0]);
    outfile = fopen("a.out","wb");
    fwrite(buffer, 4, 1, outfile);
    fclose(outfile);
    return 0;
}
```

exp1_3_array.cpp

```
#include "stdio.h"
int main()
{
    int *a, n, i, t;
    scanf("%d",&n);
    a = new int [n];
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=0;i<n/2;i++){
        t = a[i];
```

```
        a[i] = a[n-i-1];
        a[n-i-1] = t;
    }
    for(i=0;i<n;i++) printf("%d ",a[i]);
    return 0;
}
```

exp2_1.cpp 动态规划-自底向上

```
#include <stdio.h>

typedef struct{
    int weight;
    int value;
}Item;

int knapsack_dp_loop(int item_num, Item* item_list, int knapsack_capacity);


int main(){
    FILE* infile;
    int in_buffer[512], item_num, knapsack_capacity;
    infile = fopen("test.dat", "rb");
    fread(in_buffer, sizeof(int), 512, infile);
    fclose(infile);
    knapsack_capacity = in_buffer[0];
    item_num = in_buffer[1];
    Item* item_list = (Item*)(in_buffer + 2);
    printf("%d\n", knapsack_dp_loop(item_num, item_list, knapsack_capacity))
;
    return 0;
}

#define MAX_CAPACITY 63

int knapsack_dp_loop(int item_num, Item* item_list, int knapsack_capacity){

    int cache_ptr[MAX_CAPACITY + 1] = {0};
    for(int i = 0; i < item_num; ++i){
        int weight = item_list[i].weight;
        int val = item_list[i].value;
        for(int j = knapsack_capacity; j >= 0; --j){
            if(j >= weight){
                cache_ptr[j] =
                    (cache_ptr[j] > cache_ptr[j - weight] + val)?
```

```
                    cache_ptr[j]:
                    cache_ptr[j - weight] + val;
            }
        }
    }
    return cache_ptr[knapsack_capacity];
}
```

exp2_2.cpp 遍历搜索

```
#include <stdio.h>

typedef struct{
    int weight;
    int value;
}Item;

int knapsack_search(int item_num, Item* item_list, int knapsack_capacity);

int main(){
    FILE* infile;
    int in_buffer[512], item_num, knapsack_capacity;
    infile = fopen("test.dat", "rb");
    fread(in_buffer, sizeof(int), 512, infile);
    fclose(infile);
    knapsack_capacity = in_buffer[0];
    item_num = in_buffer[1];
    Item* item_list = (Item*)(in_buffer + 2);
    printf("%d\n", knapsack_search(item_num, item_list, knapsack_capacity));

    return 0;
}

int knapsack_search(int item_num, Item* item_list, int knapsack_capacity){
    int val_max = 0;
    for(int state_cnt = 0; state_cnt < (0x1 << item_num); ++state_cnt){
        int weight = 0;
        int val = 0;
        for(int i = 0; i < item_num; ++i){
            int flag = (state_cnt >> i) & 0x1;
            weight = flag? (weight + item_list[i].weight): weight;
            val = flag? (val + item_list[i].value): val;
        }
        if(weight <= knapsack_capacity && val > val_max)val_max = val;
```

```
    }
    return val_max;
}
```

exp2_3.cpp 动态规划-自顶向下

```
#include <stdio.h>

typedef struct{
    int weight;
    int value;
}Item;

int knapsack_dp_recursion(int item_num, Item* item_list, int knapsack_capacity);

int main(){
    FILE* infile;
    int in_buffer[512], item_num, knapsack_capacity;
    infile = fopen("test.dat", "rb");
    fread(in_buffer, sizeof(int), 512, infile);
    fclose(infile);
    knapsack_capacity = in_buffer[0];
    item_num = in_buffer[1];
    Item* item_list = (Item*)(in_buffer + 2);
    printf("%d\n", knapsack_dp_recursion(item_num, item_list, knapsack_capacity));
    return 0;
}

int knapsack_dp_recursion(int item_num, Item* item_list, int knapsack_capacity){
    if(item_num == 0)return 0;
    if(item_num == 1){
        return (knapsack_capacity >= item_list[0].weight)? item_list[0].value: 0;
    }
    int val_out = knapsack_dp_recursion(item_num - 1, item_list + 1, knapsack_capacity);
    int val_in = knapsack_dp_recursion(item_num - 1, item_list + 1, knapsack_capacity - item_list[0].weight) + item_list[0].value;
    if(knapsack_capacity < item_list[0].weight)return val_out;
    else return (val_out > val_in)? val_out: val_in;
}
```