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. 综合练习

练习 2：背包问题

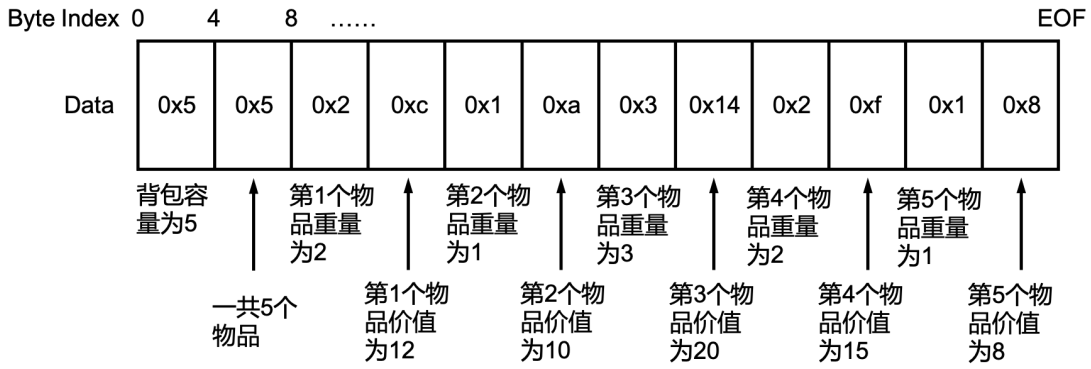
给定n个重量为$w1, w2, \cdots, wn$，价值为$v1, v2, \cdots, vn$的物品和一个承重量为W的背包，如果选择一些物品放到背包中，求使得背包中物品价值最大的方案。

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。**

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_capaci
ty);

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;
}
```