CS-382 Computer Architecture and Organization Fall 2022

Homework 1 · C Language

Lecturer: Shudong Hao *Date: See Canvas*

In this homework, we're going to implement control structures using `goto` statements. Even though `goto` is not something you should use normally in C programming, we use it here as a tool to get familiar with un-structured programs, which will benefit our upcoming assembly language learning.

1 Task 1 (20 pts): Copy String `void strcpy(char*,char*)`

In this task, you will write a C code to implement the string-copy function, that copies all the characters in the `src` string to `dst` string. This function is declared in header file `<string.h>`, but you are **not allowed** to use it. Your code also don't need to know the length of the string, so the C function `strlen()` is **not allowed**.

The prototype of the function is declared as follows:

```c
void strcpy(char* src, char* dst);
```

You can assume `dst` has enough space to store all the characters from `src`. Note, you **cannot use** any type of structured loops, such as `for`, `while`, and `do...while`, meaning the only option you can use is `goto` statement. Of course you can use `if-else` structure when needed.

1.1 Requirements

- Do not use structured loops and everything mentioned above.

2 Task 2 (30 pts): Calculate Dot Product

`int dot_prod(char*,char*,int,int)`

In this task, you need to write a C function to calculate the dot product between two vectors. The prototype of the function is declared as follows:

```c
int dot_prod(char* vec_a, char* vec_b, int length, int size_elem);
```

where both `vec_a` and `verb_b` are integer vectors of length `length`, and the function will return a single integer as the dot product. `size_elem` is the number of bytes of each element in the vectors.

Note that even though both vectors contain integer values, we do not pass `int*`. In your implementation, you shouldn't cast the entire vector back to `int*`. However, casting one element to `int*` is allowed. As in the previous task, you are not allowed to use loops, so you'd have to use `goto` statements.

2.1 Requirements

- Do not use structured loops;
- Do not cast the entire array into another type – you should only cast one address to `int*` at a time.

3 Task 3 (50 pts): Sorting Nibbles `void sort_nib(int*)`

In this task, you'll write a C function to sort all the nibbles (4 bits) in an integer array. The prototype of the function is declared as follows:

```
1 void sort_nib(int* arr);
```

For example, say we have an integer array:

```
1 int arr[3] = {0x12BFDA09, 0x9089CDBA, 0x56788910};
```

One nibble has 4 bits, so each hexadecimal digit represents a nibble. If we treat them as individual numbers and sort them from smallest to largest and print them out as integers, we have:

```
1 0x00011256  0x78889999  0xAABBCDDF
```

You can use any sorting algorithm you like, but do not use existing libraries. **10 bonus points** if you use `goto` in your code instead of structured loops. [1]

3.1 Requirements

- Do not use existing libraries to sort;

[1] Note either you use loops **or** `goto`. There's no partial bonus points for using mixed `goto` and loops.

- Write down the sorting algorithm you chose in the comments;
- In the comments state if you'd like to be graded for bonus points. Without the statement no bonus points will be given.

4 Starter Code

The following is a starter code where you can see how to call and test the two functions:

```c
#include <stdio.h>

/*
   Your name and honor code
   State the sorting algorithm you chose in task 3
   State if you want to be considered for bonus points in task 3
*/

void strcpy(char* src, char* dst) {
    /* Your code here */
}

int dot_prod(char* vec_a, char* vec_b, int length, int size_elem) {
    /* Your code here
       Do not cast the vectors directly, such as
       int* va = (int*)vec_a;
    */
}

void sort_nib(int* arr) {
    /* Your code here */
}

int main() {

    char str1[]    = "382 is the best!";
    char str2[100] = {0};

    strcpy(str1, str2);
    puts(str1);
    puts(str2);

    int vec_a[3] = {12,34,10};
    int vec_b[3] = {10,20,30};
    int dot      = dot_prod((char*)vec_a, (char*)vec_b, 3, sizeof(int));
```

```c

    printf("%d\n", dot);

    int arr[3] = {0x12BFDA09, 0x9089CDBA, 0x56788910};
    sort_nib(arr);
    for (int i = 0; i < 3; i ++) {
        printf("0x%x ", arr[i]);
    }
    puts("");

    return 0;
}
```

5 General Requirements

Code that doesn't compile will receive zero credits – no exception!

- Write your name and honor code pledge at the top of your code as comments;
- Do not change anything provided in the starter code, except in `main()` where you can write your own tests;
- You shouldn't need to create any other function or include any header files other than `<stdio.h>`;
- Comment your code well – describe what your code does. Meaningless comments and/or comment-less code will be penalized;
- When in doubt, always ask.

Deliverable

Submit a single `.c` file on Canvas.