**Task 1a: Addresses**

We read, compiled and ran the addresses.c program.

Diagram

Description automatically generatedText

Description automatically generated

Questions:

1. Could you tell the location of each memory address?

2. What can you say about the numerical values? Do they obey a particular order?

When answering these questions, we will keep in mind the typical memory arrangement

Answers:

Variables in which are declared in the stack are at the top of the memory and have an address that starts with: "0xff".

Under the stack are variables in which are declared in the Heap and have an address that starts with: "0x57".

Lastly, variables under the heap which are declared in the initialized/uninitialized data and text segment have an address that begins with: "0x56" .

In addition, we will not that when declaring a new variable it is declared above the old one and the space which is between them is based on the size of the declared type.

Therefore, for the following output we got:

- &addr2: 0xff924f2c ***Stack Address***

- &addr3: 0xff924f30 ***Stack Address***

- foo: 0x565a56e0 ***Text Segment Address*** //text segment->function declaration

- &addr5: 0x565a7018 ***Uninitialized Data (BSS) Address***

- addr0: 0x565a7008 ***Initialized Data Address***

- addr1: 0x565a7010 ***Uninitialized Data (BSS) Address***

- &addr6: 0x565a7014 ***Uninitialized Data (BSS) Address***

- yos: 0x565a5880 ***Initialized Data Address ///*** ***// yos pointer in stack-> text segment***

- addr4: 0x57c59160 ***Heap Address //addr4 pointer in stack -> heap***

- &addr4: 0xff924f34 ***Stack Address (reference)***

**Task 1b: Distances**

The purpose of the distances printed in the point\_at function is  
Where is each memory address allocated and what does it have to do with the printed distance?

This is the output we got:

dist1: (size\_t)&addr6 - (size\_t)p: -4

dist2: (size\_t)&local - (size\_t)p: -1455956284

dist3: (size\_t)&foo - (size\_t)p: -6456

dist1 is the distance between the address of addr6 and addr5 which both allocated on the Uninitialized Data (BSS).

The distance equals -4 since addr5 is of type int which is 4 bytes.

dist2 is the distance between the address of local and addr5 which local is allocated in the Stack and addr5 in the Uninitialized Data (BSS).The distance equals -1450185116 since Uninitialized Data (BSS) variables are located under the Heap, which begin far from the Stack at the top.

dist3 is the distance between the address of foo and addr5 , such that f is allocated in the Text Segment and addr5 is in the Uninitialized Data (BSS).The distance equals -6451 since Uninitialized Data (BSS) variables are above the Initialized Data, which is above the Data Segment.

**Task 1c: Arrays Memory Layout**

1. for the output:

- int array value at index 0: 0xffea4e8c

- int array value at index 1: 0xffea4e90

- int array value at index 2: 0xffea4e94

- char array value at index 0: 0xffea4e99

- char array value at index 1: 0xffea4e9a

- char array value at index 2: 0xffea4e9b

- iarray: 0xffea4e8c

- iarray+1: 0xffea4e90

- carray: 0xffea4e99

- carray+1: 0xffea4e9a

2. The behavior of the '+' operator is to move the pointer to point on the next address where the next variable is declared in the memory.

This jump is based on the size of the type of variable we are jumping to.

In our case, for int variables we jumped 4 spaces (4 bytes), and for char variables we jumped 1 space (1 byte).