David Hatcher

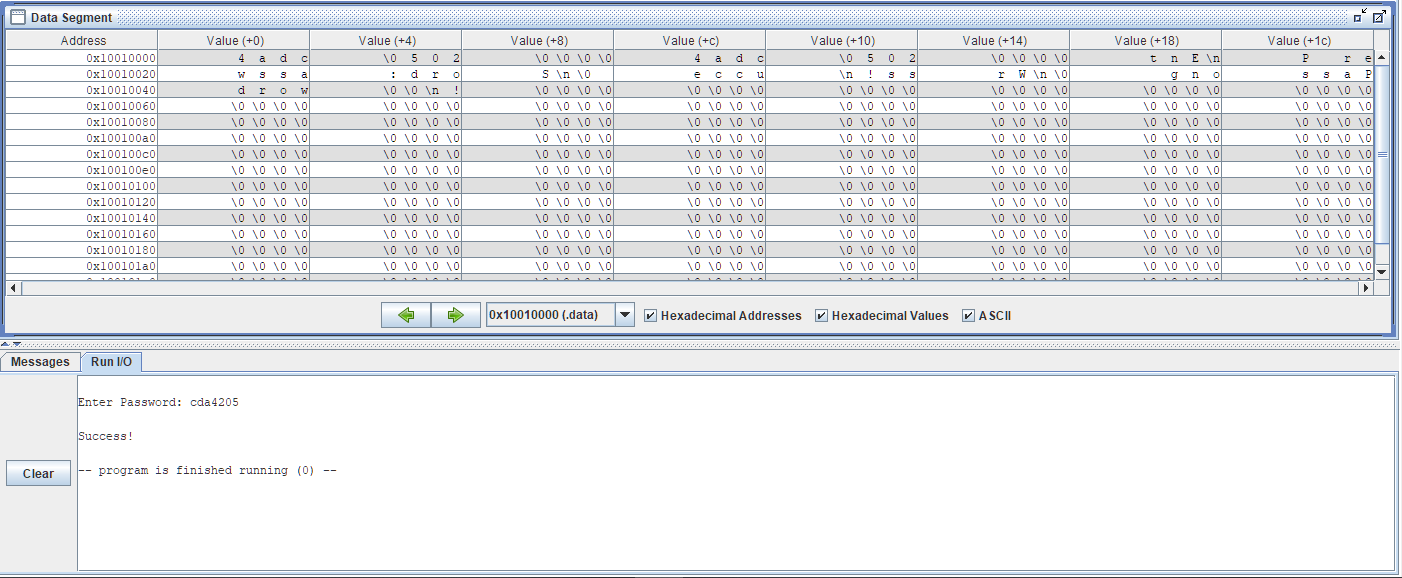
CDA4205

Homework 4

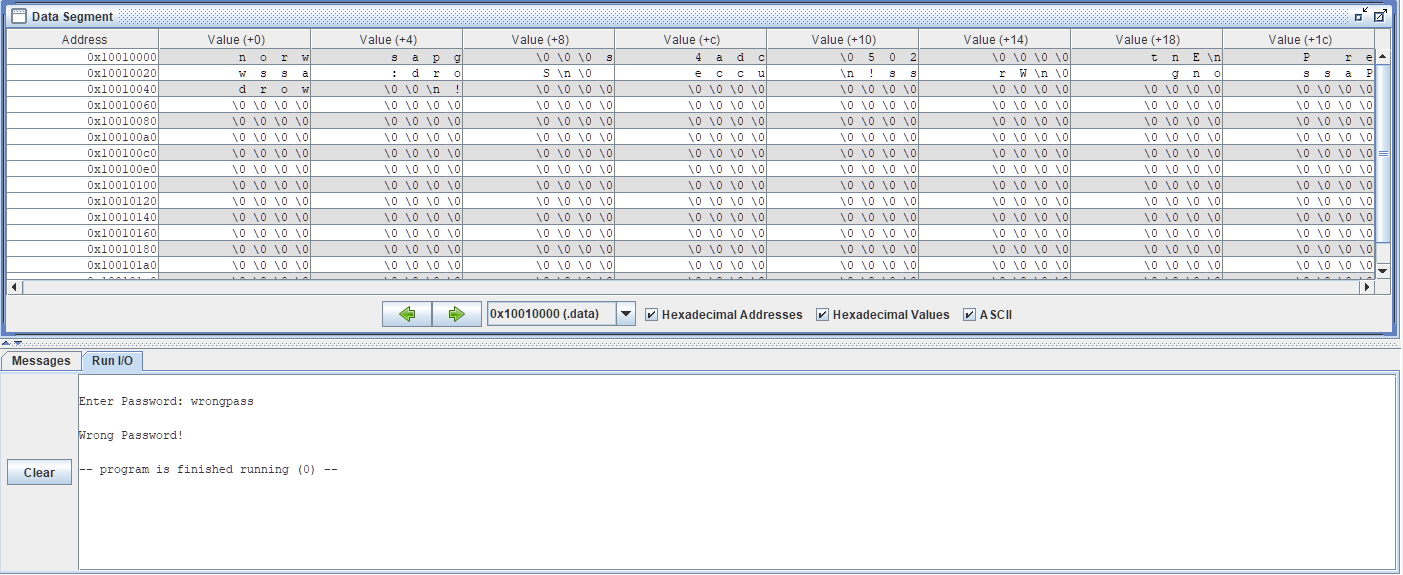
1. The stack pointer allows the program to return to the point in memory that it was at before the procedure was called.
2. A sequence of instructions without branches and without branch targets or branch labels. The compiler creates these by breaking the code up into the basic blocks.
3. Heap is dynamic data, stack is automatic storage. The stack is managed by the system and the heap is managed by the user. The stack is used for memory allocated during compilation, an example could be a fixed length array, and the heap is used more memory allocated during run-time, an example could be a linked list.
4. This is an important design choice because it increases efficiency and ease of development for the user. The impact this has on RISC-V systems is that since the instruction format is consistent throughout all of the instructions the this allows the decoder to just look for the opcode where it always is and decode the instruction rather than look over the entire instruction to determine what it needs to do with it.
5. The compiler turns code from a high-level human readable language into an assembly language program. The assembler takes the assembly language program and translates it to machine instructions. The linker takes the machine instructions and combines it with libraries that were used and turns it into an executable image. The loader then takes the executable image and places it into the memory, initializes all the registers and sets up all the memory the program will need, then jumps to the startup routine.
6. Static linking is where the linking of external libraries is done when the program is compiled, and Dynamic linking is where the libraries are linked when the procedure is called. The pros of Static Linking is that it is the speed of calling a statically linked library is much faster than a dynamically linked library and if an updated version of the library would do something that will break the way the code runs then it will not affect a Statically linked library as it’s already been compiled with the old version. The pros of Dynamically linked libraries is that any updates will be automatically used whenever the library is called and it avoids a lot of the image bloat. The cons for Statically Linked Libraries are that the executable image is much bigger as the entire library has to be included in the code and that newer versions of the library code will not be used unless the program is recompiled. The cons for Dynamically Linked Libraries is that it could be slower as it has to go get those libraries during runtime rather than compile time. The main requirement for Dynamically Linked Libraries is the procedure code must be relocatable.

**Programming Assignment Screenshots**

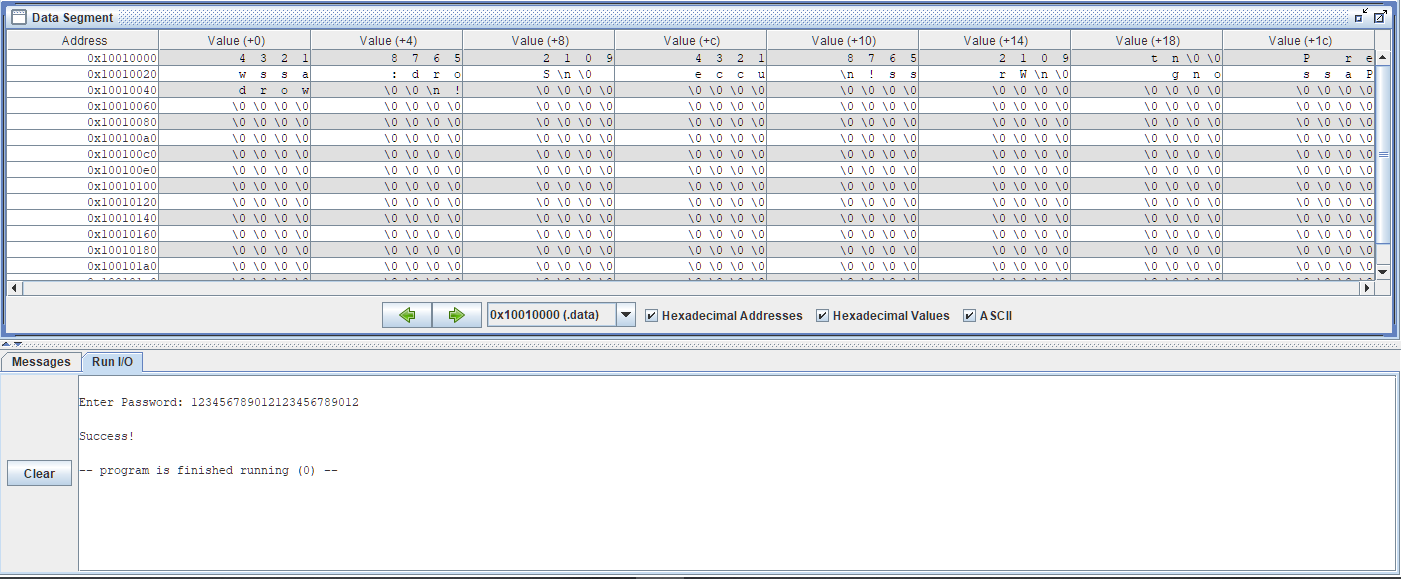
**Success**

****

**Failure**

****

**Buffer Overflow Attack**

****

1. You can enter a sequence of repeating characters and still log in with the wrong password, you do not need to just enter the same repeating character. In the previous screen you can see that I entered the string 123456789012123456789012 and it accepted it as the correct password.
2. You need to input at least 24 characters. This is because the size of both of the reserved memory locations for the correct password and the password entered are both 12 bytes long. Entering 24 characters causes both of them to be overwritten and make the first 12 characters match to the second 12 characters.