GUTIERREZ, JETER TAKE HOME TEST 1 README FILE

TESTING VISUAL STUDIO WINDOWS 32 BIT COMPILER EXAMPLES.
Visual Studio project files are cpp files and are located within the directory GUTIERREZ_JETER_CSC_343_FALL_2017_TAKE_HOME_TEST_1/TAKE HOME TEST C/WINDOWS/GUTIERREZ_example/example/example where example is the name of the example that is being tested. There are 8 examples that are going to be tested. Open each of the .CPP files using Visual Studio in order to run the test.
PROCEDURE:

- First we will need to build the solution in order to start debugging each of the examples that we are testing. After we build the solution we will step into the program by pressing F10 and continually do that until reaching the end of the program. As we do that we will also be observing the windows provided to us through Visual Studio. The windows that we will look for changes in are the memory windows, assembly window and registers window, in the memory window we will set the address of the current line in memory displayed equal to the address of the base pointer.

MIPS EXAMPLES ON MARS 32 BIT SIMULATOR.
The 7 MIPS examples are located in the directory GUTIERREZ_JETER_CSC_343_FALL_2017_TAKE_HOME_TEST_1/TAKE HOME TEST Mips/. The MARS4_5.jar file is also provided and can be opened to test the MIPS examples. We will open the 7 MIPS examples directly from the MARS simulator and do the following for each example:

- We will assemble the program by finding the menu bar that is located in the top of the simulator program and then Run → Assemble. We will step into the program by pressing F7 when we do we will now have an updated data segment and text segment. In the data segment we can see the addresses that are used in the Stack as well as the information that is being stored in those locations. In the registers windows we can see the 32 registers that will be used and updated as we run through the program that is in assembly code in the text segment window. We will continue to press F7 in order to finish running through the end of the program and once we are finished we can simply open the next example.

LINUX 64 BIT GCC AND GDB EXAMPLES ON UBUNTU AND LINUX 32 BIT COMPILER GCC GDB EXAMPLES ON CORTEX ON THE RASPBERRY PI.
The files that will be tested in both Linux platforms are written in C and are in the directory GUTIERREZ_JETER_CSC_343_FALL_2017_TAKE_HOME_TEST_1/TAKE HOME TEST C/GUTIERREZ_LINUX_AND_PI. Before we begin debugging in Linux we will first need to

compile and link the program in order to make it usable with GDB. we will us GCC compiler and the following command in order to compile and link:

- cc -g -o0 example.c -o example_test

In the previous line example will be replaced with the name of the example that is going to be tested.

Now that we have compiled and linked our examples it is time to begin the debugging process in Linux which can simply be ran with the following command that launches GDB debugger:

- gdb example_test

Once we have launched GDB we will have to do the following in order to debug the program:

- We need to type in *break main* in order to set the breakpoint for the debugger, after that we need to type *run* so the program can begin and reach our breakpoint. At this step we can type *disassemble* in order to see the assembly code and information that is happening in the background when we are using GDB and running our examples. In order to continue in the program instead of pressing F7 like in MIPS or F10 like in visual studio we will simply continue to write *next 1* until reaching the end of the program or by simply typing in *q* to exit and *y* when prompted to confirm exiting the debugger.
- We can also print out values and addresses that we want to see when using the GDB debugger. In order to print the address of the base pointer on Linux 64 bit we will simply type *x $rbp* and on the raspberry pi we will type *x $r11*. To see the value and address of a variable we simply write in either case, assuming for example we have a variable named Local_Variable we would type *x &Local_Variable*. Then we exit the program by either closing the terminal or typing *q* followed by *y* if prompted.