CSSE 132 – Introduction to Systems Programming
Rose-Hulman Institute of Technology

Lab 3 – Question Sheet

For each of these questions, clearly explain your answers and write out any commands the questions request.

Name (Print):______________________ RHIT Username:______________

Part 1: GDB

1. What is the difference between the commands `next` and `nexti` (or `n` and `ni`).

   `n` is for next line of [ ] but `ni` is for next [ ].

2. What is the difference between the commands `next` and `step` (or `n` and `s`).

   `n` is for stepping [ ] functions but `s` stepping [ ] functions if any encountered.

3. What is the command to set a break point at line `X`?

   [ ]

4. What is the command to print out the value of a register named `X`?

   [ ]

5. What is the command to run the program to the next break point?

   [ ]

Part 2: Stack/Procedure Call - Local Variables

Read the web instructions for this part before continue.

6. (**Breakpoint: Line 9**) At the beginning of `main`, what is the value stored in the register `sp`? (*You can either print the register or check the register panel*)
   Fill your answer in the cell of Table **??** (on the next page) where shows the label $\boxed{\texttt{C1}}$.

   (Do not fill any other cells for now.)

7. (**Breakpoint: Line 11**) After assigning of the values of `a` and `b` in `main`, check out the *stack* panel in GEF (scroll up in GEF to find the *stack* panel), where do you think the local variables `a` and `b` in `main` are stored in memory. (Check the source code to see the values of the variables and find those numbers on the stack)

   Fill your answers in Table **??** (on the next page) into the corresponding cells with labels indicated below.

   Put the relative address of `a` (i.e., `sp + N`) in $\boxed{\texttt{C2}}$, and the value in $\boxed{\texttt{C3}}$.

   Put the absolute address of `b` in $\boxed{\texttt{C4}}$, the relative address (i.e., `sp + N`) in $\boxed{\texttt{C5}}$, and the value in $\boxed{\texttt{C6}}$.

8. (**Breakpoint: Line 4**) At the beginning of `do_nothing`, what is the value stored in the register `sp`?

   Fill your answer in the cell of Table **??** (on the next page) where shows the label $\boxed{\texttt{C7}}$. *(Do not fill any other cells for now.)*

9. (**Breakpoint: Line 6**) At the end of `do_nothing`, check the *stack* panel in GEF (scroll up in GEF to find the *stack* panel), where do you think the local variables `a` and `b` in `do_nothing` are stored in memory. (Check the source code to see the values of the variables and find those numbers on the stack)

   Fill your answers in Table **??** (on the next page) into the corresponding cells with labels indicated below.

   Put the relative address of `a` (i.e., `sp + N`) in $\boxed{\texttt{C8}}$, and the value in $\boxed{\texttt{C9}}$.

   Put the absolute address of `b` in $\boxed{\texttt{C10}}$, the relative address (i.e., `sp + N`) in $\boxed{\texttt{C11}}$, and the value in $\boxed{\texttt{C12}}$.

10. (**Breakpoint: Line 12**) At the end of `main`, what is the value stored in the register `sp`?

    Find out this value in GEF and check with Table **??**, what is the current value `sp` equivalent to? Circle the right answer below.

    $\boxed{\texttt{sp}_{\,main}}$ $\boxed{\texttt{sp}_{\,do\_nothing}}$

Absolute Addr	Relative Addr	Data	Variable	
0x C4	C5	C6	b	
0x C1	C2	C3	a	← sp $_{main}$
0x C10	C11	C12	b	
0x C7	C8	C9	a	← sp $_{do\_nothing}$

Table 1: Stack for Part 2 `do_nothing.c`

11. Based on the observations above, you should have some rough ideas of how local variables are handled in a program. Here is a summary:
    Each function has its own private space to store its local variables. This space is called *Stack Frame*. Namely, each function has its own *stack frame*.

    The register `sp` is the *stack pointer*. To find out how it works, we need to dive into the Assembly code. Now open `do_nothing.s` (type `make do_nothing.s` to generate it), carefully check `do_nothing` function and find the instructions that modify `sp`.

    - What is the number that is subtracted from `sp` at the beginning of `do_nothing` function?
    - Why do you think it is this particular value? Think about what data/variables are stored in this function.

12. Overall, what is the purpose of `sp`?
    It points to (i.e., stores the address of) the stack frame of the
    first previous current last
    function. (Circle the one correct answer above)

Ask your instructor to verify your answers and sign off.

Verified:________________________ Date/Time________________

Part 3: Stack/Procedure Call - Function Input Arguments

Read the web instructions for this part before continue.

13. (**Breakpoint: Line 12)** Right before calling function `add`, check the ARM panel in GEF, there are the two `ldr` instructions before the `bl` instruction (which will branch to the `add` function).

```
ldr r1, [sp, #4]
ldr r0, [sp]
```

Step over these two instructions (using `ni`) to see what they try to do?

They assign the value of $\boxed{}$(answer with a variable name in C), which is $\boxed{}$(answer with a scalar number), to register $\boxed{}$, and the value of $\boxed{}$(answer with a variable name in C), which is $\boxed{}$(answer with a scalar number), to register $\boxed{}$

14. (**Breakpoint: Line 4)** At the beginning of `add` before running `int c = 8`, check the ARM panel in GEF, there are the two `str` instructions:

```
str r0, [sp, #4]
str r1, [sp]
```

What are these two instructions trying to do?

They assign the value of register $\boxed{}$, which is $\boxed{}$(answer with a scalar number), to the input argument $\boxed{}$ (answer with a variable name in C), and the value of register $\boxed{}$, which is $\boxed{}$(answer with a scalar number), to the input argument $\boxed{}$ (answer with a variable name in C)

15. (**Breakpoint: Line 7)** At the end of `add`, check the stack panel and also the C code, complete the empty cells in the table below (*Note: Each cell is 4 bytes).*

Absolute Addr	Relative Addr	Data	Variable Name in C	
0x				
0x				
0x				
0x				← sp

Table 2: Stack frame of `add` function in Part 3

16. (**Breakpoint: Line 7**) At the end of `add`, check the ARM panel, which variable (answer with a variable name in C) is loaded to `r0`? What is the purpose of doing this?

17. Based on the observations above, explain that 1) how input arguments are passed to a function and 2) how to return a value from the function.

    To pass input arguments to a function, before calling that function, the values of input arguments are first loaded into ☐ (find the answer in Problem **??**). After going into that function, those values are further assigned to the ☐ variables (*Hint: Think about what kind of variables* `a` *and* `b` *are*).

    The return value is saved to register ☐.

18. What are the differences and similarities between *local variables* and *input arguments*?

    They are both stored on ☐, but the values of *input arguments* are assigned by ☐.

Ask your instructor to verify your answers and sign off.

Verified:____________________ Date/Time____________

Part 4: Stack/Procedure Call - Function Return

Read the web instructions for this part before continue.

19. (**Breakpoint: Line 17**) Right before calling the `add` function in `main`, in the ARM panel in GEF, you can see a `bl` instruction will be called soon. On my computer, it shows as this (your address might appear differently)

```
    bl 0xfefde524 <add>
```

    This is the instruction that will call `add` function. Now, I want to you do something seemingly odd: Write down the `address` of the next instruction in memory after this `bl` instruction.

    Hint: 1) It should be a `str` *instruction. 2) The address of an instruction is in the leftmost column in GEF (Check Part1 instruction - ARM Panel for reference)*

    ☐

20. **(Breakpoint: Line 8)** At the beginning of the `add` function,

    - What is the value stored in the register `$lr` at this moment?
    - Compare your answer above to your answer to Problem **??** (*Hint: They should be the same)*, what is this address (related to function `add`)? Why does it matter?

21. **(Breakpoint: Line 10)** Right before calling the `do_nothing` function in `add`, in the ARM panel in GEF, you can see a `bl` instruction will be called soon. This is the instruction that will call `do_nothing` function. Similar to Problem **??**, write down the `address` of the next instruction in memory after this `bl` instruction.

    Hint: It should be a `ldr` *instruction.*

22. **(Breakpoint: Line 4)** At the beginning of the `do_nothing` function, what is the value stored in the register `$lr` at this moment? *(It will be different from the answer to Problem* **??***)*

23. **(Breakpoint: Line 5)** At the end of the `do_nothing` function, in the ARM panel in GEF, you can see this instruction `bx lr` will be run soon. Given that `bx` is almost equivalent to `b`, what do you think will happen after running this instruction?

    It will go back to ______ function, specifically in line ______ (write a line number in C).

24. Based on the observations above, what do you think the instruction `bl` does? Assume the specific instruction is the one in Problem **??**

```
    bl  0xfefde524 <add>
```

(*Hint: It changes the register* `$lr`.)

The instruction will do two things: 1) Branch to ______ (which is the beginning of function ______)

2) Store the address of the ______ instruction to `$lr`. (Fill in a word that describes the position w.r.t to the `bl` instruction)

Ask your instructor to verify your answers and sign off before turning in this sheet.

Verified:______________________ Date/Time______________