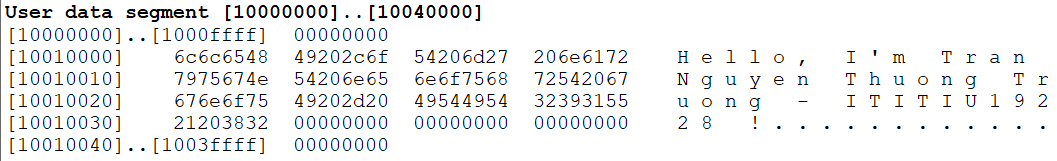
ITITIU19228 – TRẦN NGUYỄN THƯƠNG TRƯỜNG – REPORT COMPUTER ARCHITECTURE LAB 1

**Question 3.1:**

* The memory address of an instruction in hexadecimal .



**Question 3.2:**

* When stepping through the program, the next instruction to be executed is highlighted instruction and thus has not yet been executed. As such, that instruction has not yet been executed. This highlighting is how to track the progress of the program execution.

**Question 3.3:**

* In string “Hello, MIPS\n”, attempt location memory to execute non-instruction at 0x00400030.
* Look into Data Window the content of the memory segment whicstores that string (in hex) is:

**User data segment [10000000]..[10040000]**

[10000000]..[1000ffff]  00000000  
[10010000]    6c6c6548  57202c6f  646c726f  00000a21

[10010010]..[1003ffff]  00000000

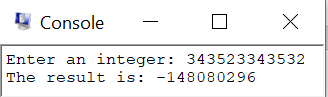
**Question 3.4:**

* The MIPS code to print out my full name and student ID on separate lines by **name.s** file.

Graphical user interface, text

Description automatically generated

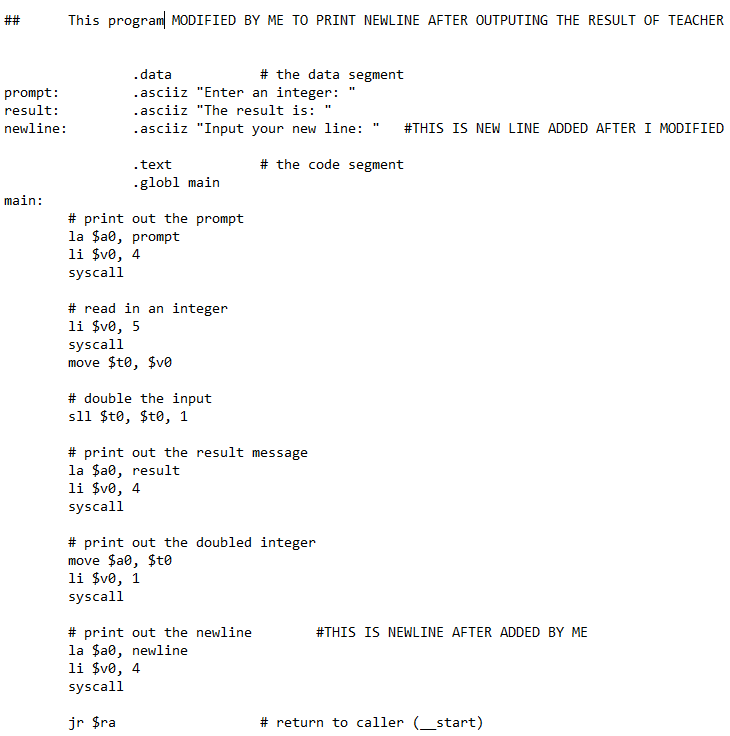
**Question 4.1:**

* The output when i enter the integer ***343523343532*** is: -148080296 
* Output is incorrect. Because, in program ***lab1\_integer\_double.s*** we set values input is Integer number type. But Integer number type have value from -215 to 215 -1, meaning from -32768- 32767. So, when we enter the integer ***343523343532,*** its value is so big lead to exceed integer type and compiler will choose random number in value range of integer type to output.

**Question 4.2:**

* Maximum and minimum input values that program can calculate correctly must be in range (-32768- 32767).

**Question 4.3:**

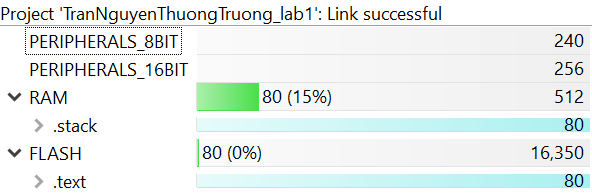
* 

**Question 5:**

* This MIPS code i saved in **lab1\_twointeger.s** file. Please check it hepl me.
* **Step 1:**

Fill in the percentage of RAM allocation: \_\_\_80(15%)\_\_\_\_\_ and files:

Fill in the percentage of FLASH allocation: \_\_\_64(0%)\_\_\_\_ and files:



* **Step 2:**

Please check help me in Step 2 – Question II.png file. This here !

Text

Description automatically generated with medium confidence

* **Step 3:**

address: 0000c02a

In Section Allocation MAP:

-text: 0000c000

-stack: 000003b0

-WDT: 0000fff4

-Port 1: 0000ffe4

-Reset: 0000fffe

Memory Configuration: INT02: 0000ffe4

Global symbols: sorted alphabetically by Name:

-P1IN:\_ 00000020

-P1OUT: 00000021

- main:\_ 0000c000

-WDTCL: 00000120

-LoopCtr: \_\_\_\_\_\_\_\_\_\_

- Have all **125 symbols** (line 156- 280)

* **Step 4:** The Value of registers:

P1IN: before \_\_00000020\_\_\_ and after \_\_00000020\_\_

P1OUT: before \_\_00000021 and after \_\_00000021

LoopCtr: before \_\_\_\_\_\_\_\_\_\_ and after \_\_\_\_\_\_\_\_\_\_\_

*Hint: you can use Expression or open the Core Registers tab* **(Port\_1\_2)** *to find the solution*

**Explain the values of P1IN and P1OUT registers in Port\_1\_2: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* **Step 5:** When running and pausing, **observe** the Core Registers in Registers tab and **please write down** comments about the transition of these registers in there.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* **Step 6:** how to use breakpoint in CCS : **Alt+Shift+Q,B**

**I press Alt + Shift + Q,**

1. Set your breakpoint at the desired location.

2. Select Debug → Breakpoints. The Breakpoint Manager window appears.

3. Select your breakpoint row, then right click and select Properties from the context menu. This will open the Properties Manager window for the selected breakpoint.

4. Enter your GEL expression in the Condition field in the Property Manager. This condition will be evaluated by GEL, and if it returns false, the breakpoint will not trigger and the target will be re-run. This ensures that the breakpoint is only enabled if your condition is met. Note that if the expression calls a GEL function, it must be defined in one of your already loaded GEL files. If you don’t do this, you will receive an error message that the function is not defined.

5. Click OK.

* **Step 7:** When running and pausing, click View and open ***Disassenbly*** window, write down **these instructions of sample code:**