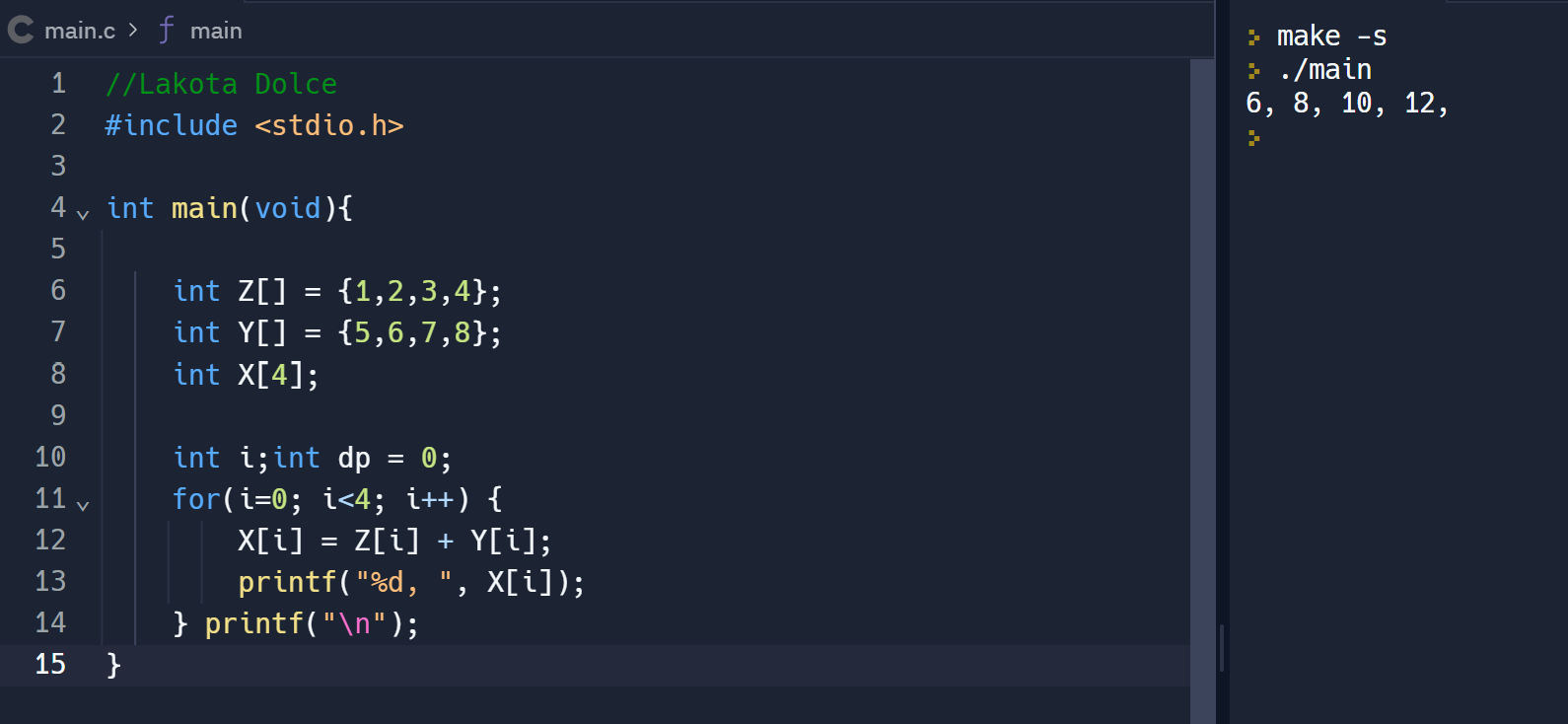
Lakota Dolce

CSCI 210

Dr. Jones

October 2, 2023

Part 1:



The picture above shows the program that I created on replit.com. To show that I wrote it myself I have changed the variables from A,B,C, to Z,Y,X, as well as added a comment with my name at the top.

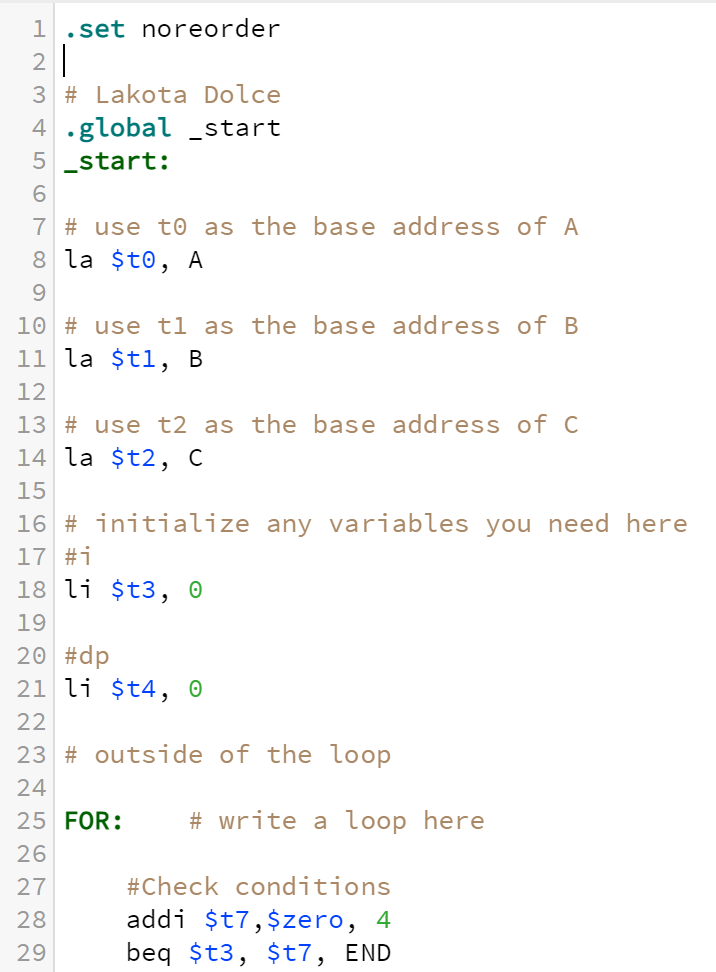
This program takes and initializes the arrays Z,Y,X with the following numbers, Z holds, 1,2,3,4 ; Y holds 5,6,7,8; and X is an empty array with 4 spaces. Next the program initializes the variables i and dp setting to 0. This is to create a parameter for the loop statement to reach and know when to stop.

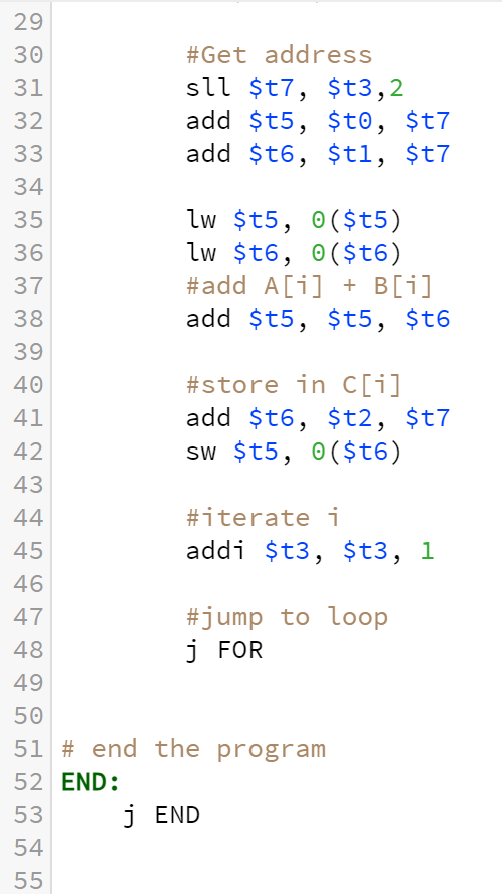
Next you have the for loop making the parameters of i starts at 0, then it will compare i and see if it is less than 4, then it wants i to go up by one number each time the loop is ran until i is no longer less than 4.

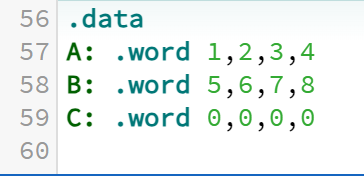
The next part uses i to iterate through the Z, and Y arrays to get a specific number out of them, then add those numbers together and store it in a specific spot in the X array. For example, the first iteration would pull the 1 from Z, and the 5 from the Y arrays, it would then add them to be 6, and store that in the first part of the array for X as i is 0 and the first piece of an array starts at 0. The next time it would pull the second number from the two arrays and store the product in the second spot of X and so on, until i has reached the number 4 breaking the loop.

The two print statements are to display the C array in the first statement, which you see in the right hand side, and the next prints an empty line for better readability after each run or for follow on code added later.

Part 2:



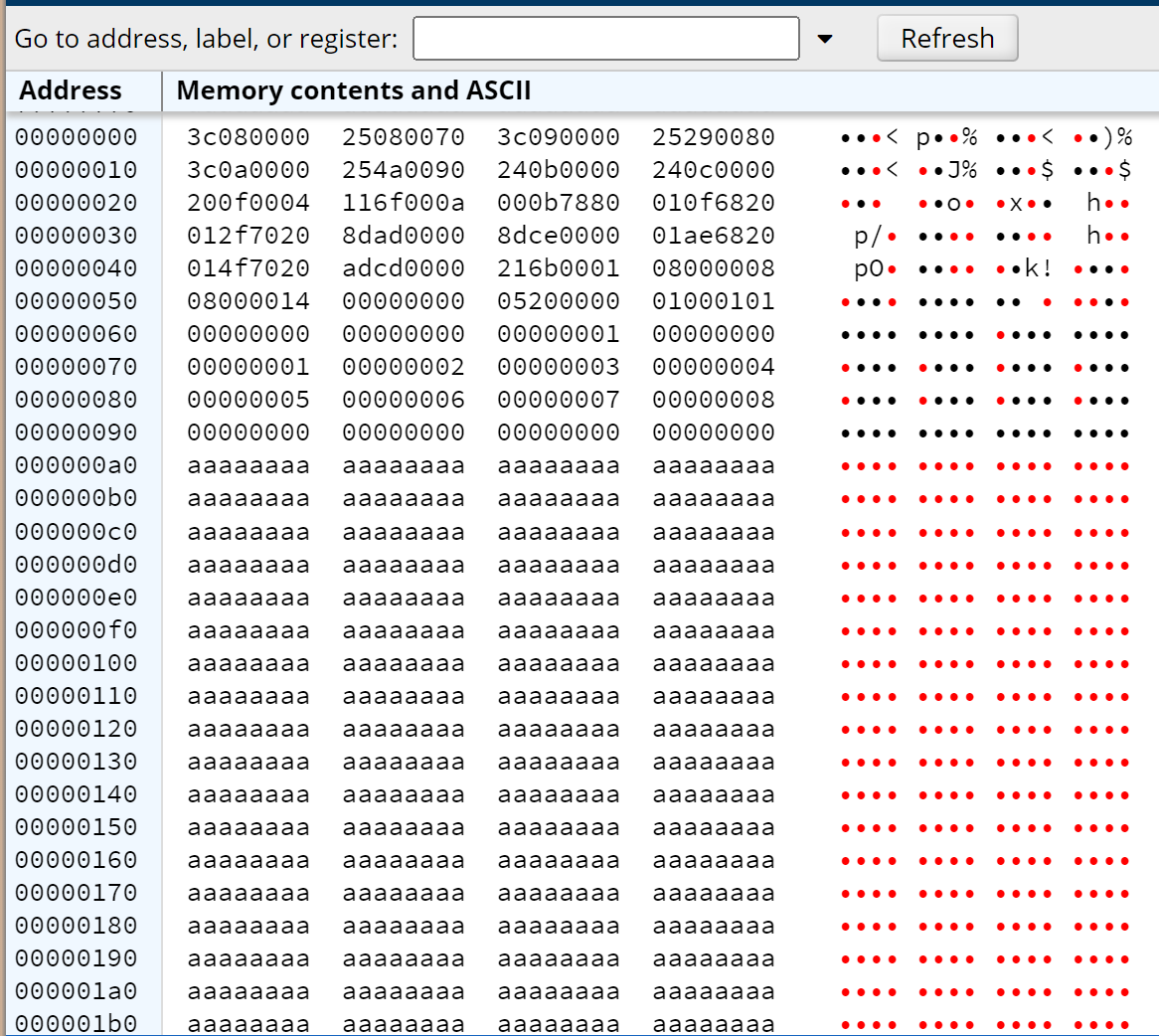




Here is the MIPS Assembly code that I wrote to accomplish the same thing as the C program without the print. I have added comments to help clarify while following the outline provided. At the top you see the start of the code, first the code saves the addresses of the arrays (shown in the last picture from above) to a the registers $t0-$t2. This is used to access this information later in the code. After the addresses are saved next the variables are initialized and store in the registers $t3 (i from the C program) and $t4 (dp from the C program).

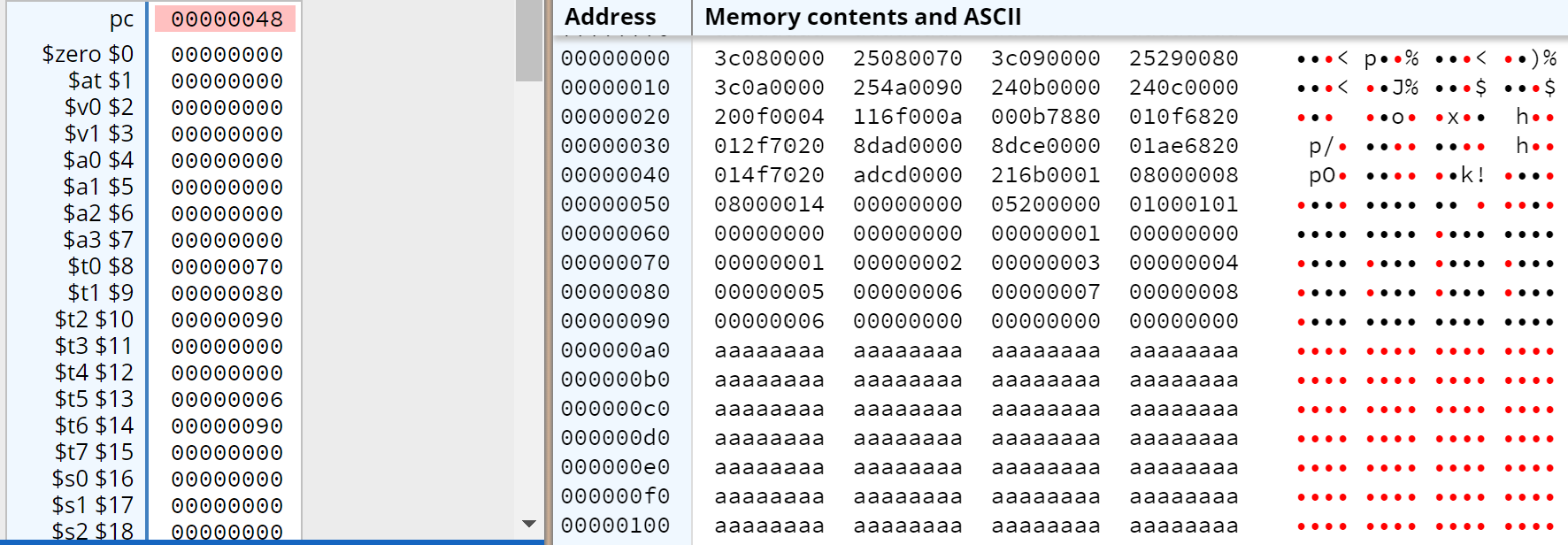
Next is the loop, at the top it creates the variable to compare i to and stores it in register $t7. It then compares if i is less than 4 and if so it continues the loop. Here we have to iterate into the array and get the variables we need to add, so we get the addresses and then use load word to get the content at that address. We take those two numbers and add them together, and then we get the address we want to store the new number at and we use store word to plant that data in the C array.

The program then adds one to i and repeats the loop until i reaches 4. Once i reaches 4 it jumps to the program end and stops the program.

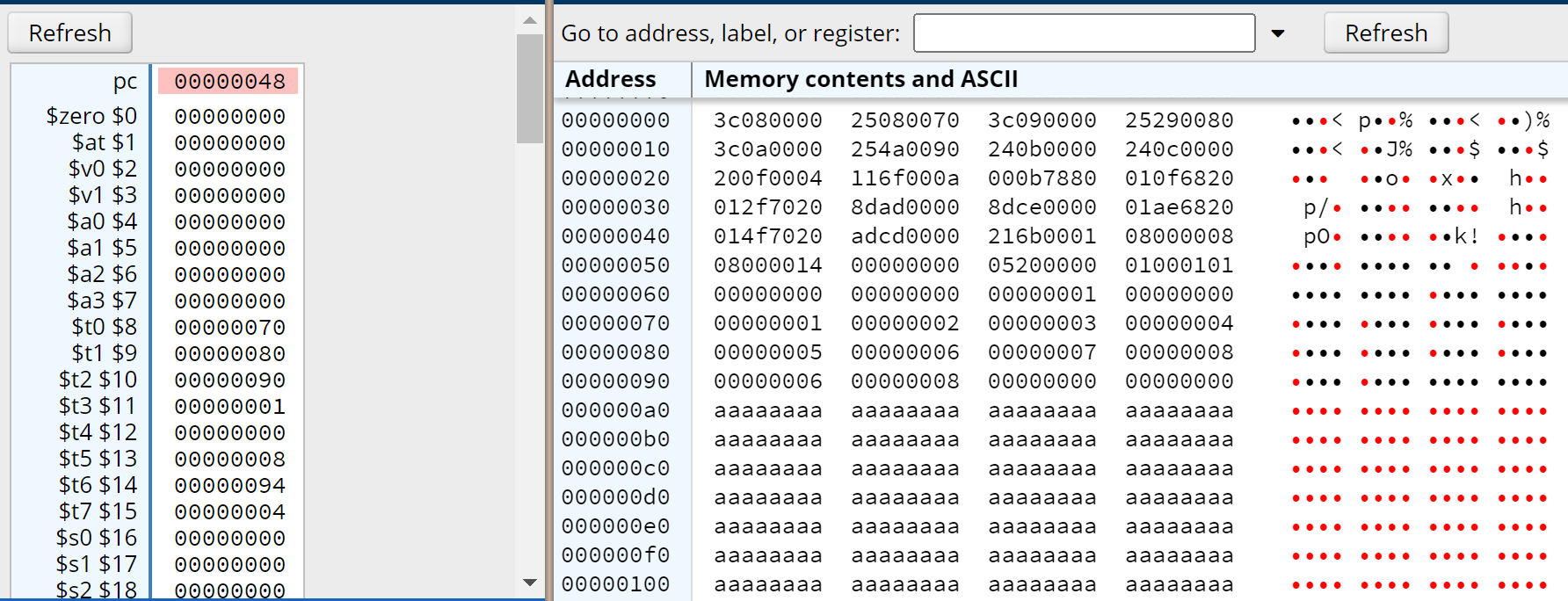




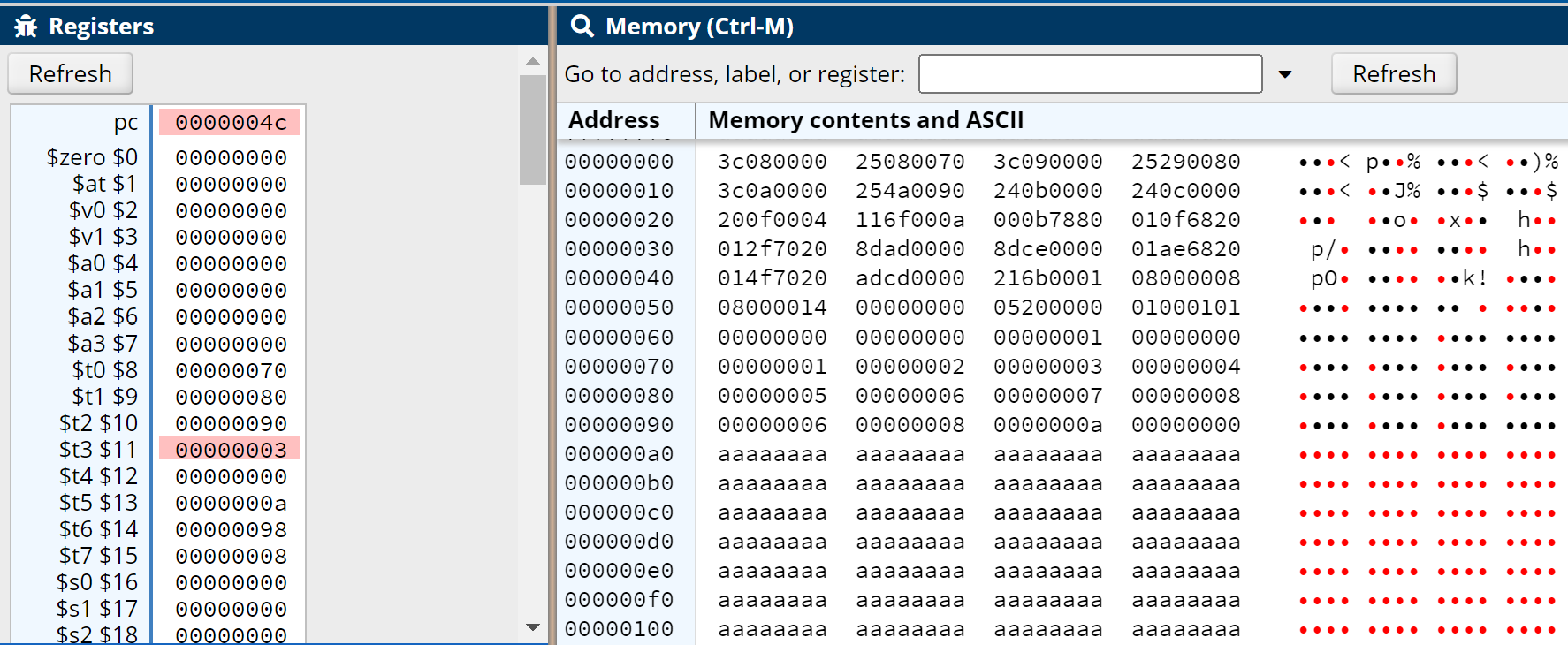
This is the memory when the program is initialized you can see that the arrays are stored in the memory at addresses, 0x070 in hex through 0x09C. We know this because the .data is stored after the programs code. Array A is 0x070-0x07C (Green), B is 0x080 – 0x08C (Red), and C is 0x090-0x09C(Blue). Further proof shows as the program runs.

  
Here is after the first iteration, you can see that registers $t0-$t2 on the left side each hold the address for the beginning of the arrays (0x070,0x080,0x090). Then register $t3 is zero for the first iteration, $t4 was initialized at 0 before the loop and will not change through the rest of the pictures, $t5 holds the sum of the two first parts of the arrays A and B and $t6 holds the address where it then stored the 6 (0x090). Next the loop repeats.







Now you see that register $t3 is now at one(purple), and the sum of the second items in arrays A and B (8) is now stored in the second address in $t6 (0x094). This continues until $t3 (i) reaches 4 where the loop breaks.  A screenshot of a computer

Description automatically generated In the End it results in the C array (addresses 0x090-0x09C) being the sums of the arrays A and B as they are iterated through them, giving us 6,8,A(hex for 10), and C(hex for 12).

