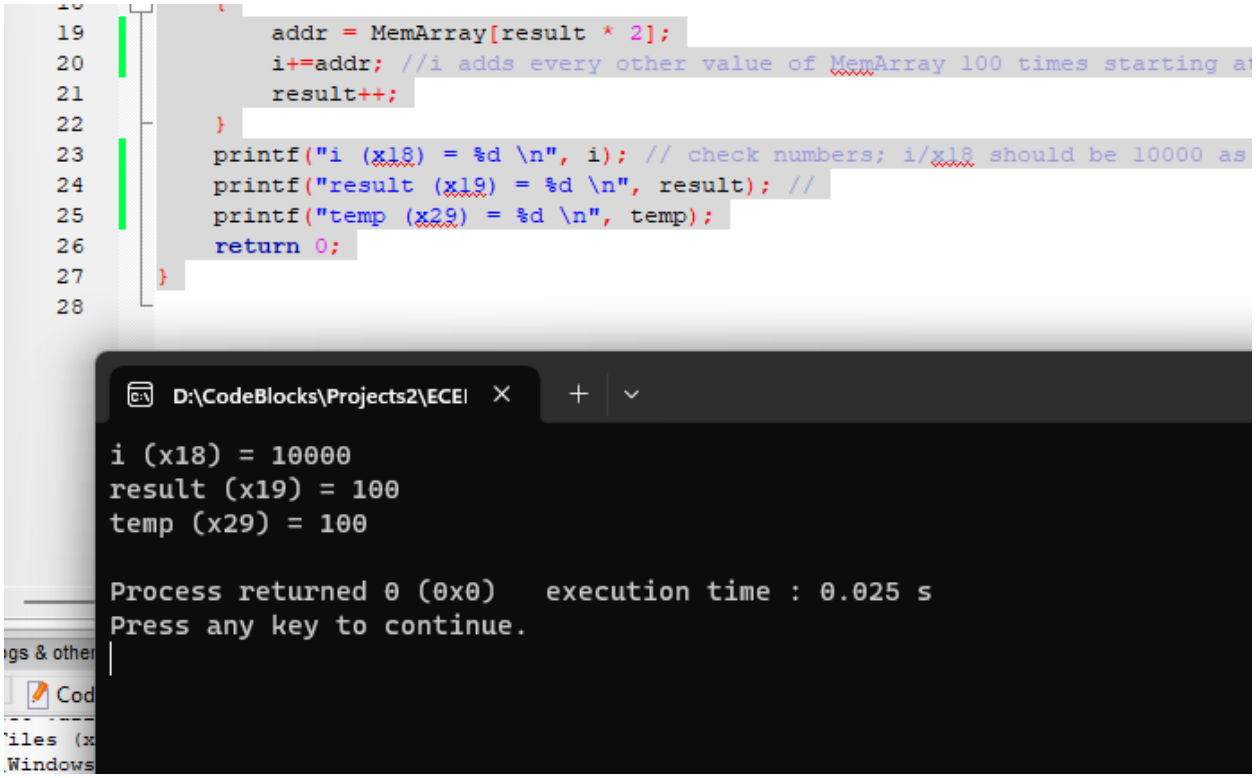
Problem 1 Code:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int i; //x18
   int result = 0;; //x19
   int MemArray [200]; //x10 and set it up with 200 numbers so it adds every other number 100
times
   for (i=0; i < 200; i++) //THIS LOOP IS ONLY TO FILL UP THE ARRAY
   {
      MemArray[i] = 100;
   }
   i = 0;
   int temp = 100; //x29
   int addr = 0; //x7 with base addr

   while (result < temp)
   {
      addr = MemArray[result * 2];
      i+=addr; //i adds every other value of MemArray 100 times starting at index 0
      result++;
   }
   printf("i (x18) = %d \n", i); // check numbers; i/x18 should be 10000 as it adds up 100 100
times
   printf("result (x19) = %d \n", result); //
   printf("temp (x29) = %d \n", temp);
   return 0;
}
```



2.

```
addi    sp,sp,-12
Sw    x20, 8(sp)
Sw x5, 4(sp)
Slli  x5, x11, 1
Add x20, x10, x5
Addi x10, x20, 0
Lw x5, 4(sp)
Lw x20, 8(sp)
Addi sp, sp, 12
jalr      x10, x1, 0
```

Note: this didn't work in the interpreter even though it should have (the interpreter sucks)

A. (0.3 pts) Explain why RISC results in a simpler instruction fetch implementation compared to CISC.

RISC results in a simpler instruction fetch implementation because of its fixed length instruction sets, single cycle execution, the instructions themselves are smaller and more focused when compared to CISC, and RISC assumes the compiler will optimize code for it.

B. (0.2 pts) Briefly explain the role of the PC.
The program counter shows/stores the address of the next instruction to be performed.
C. (0.3 pts) Briefly explain the role of Adder 1 and Adder 2 and state an instruction example that utilizes Adder 2.

Adder 1 advances the program counter as it's the instruction + 4, and adder two allows for branching as you have the instruction + some defined number to jump to. Blt x19, x29, -16 uses adder 2.

D. (0.3 pts) State an instruction example that utilizes sig_a and sig_b simultaneously.
sw x19, 0(x20): we do this because sig_a is used in store and we get the second register from sig_b which is going toward the memory in the diagram.
E. (0.3 pts) Excluding R-type instructions, state an instruction example that utilizes sig_a and sig_c simultaneously.
Addi x19, x19, 0x01: we do this because sig_a is used in the addi operation and we get the immediate from sig_c