

1 Getting Started with MARS

Download the MARS simulator v4.5 at

<https://courses.missouristate.edu/KenVollmar/MARS/download.htm>.



Figure 1. The download page of MARS

(On the MARS download page, select the "Download MARS" button.)

If you have no Java runtime environment, **download JDK** according to your system. In the case of Windows x64, choose the listed installer. (Note: ensure that the installation path is all in English without any Chinese characters.)

JDK 22	JDK 21	JDK 17	GraalVM for JDK 22	GraalVM for JDK 21	GraalVM for JDK 17
JDK Development Kit 22 downloads					
JDK 22 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC) .					
JDK 22 will receive updates under these terms, until September 2024, when it will be superseded by JDK 23.					
Linux	macOS	Windows			
Product/file description		File size	Download		
x64 Compressed Archive		185.52 MB	https://download.oracle.com/java/22/latest/jdk-22_windows-x64_bin.zip (sha256)		
x64 Installer		163.91 MB	https://download.oracle.com/java/22/latest/jdk-22_windows-x64_bin.exe (sha256)		
x64 MSI Installer		162.07MB	https://download.oracle.com/java/22/latest/jdk-22_windows-x64_bin.msi (sha256)		

Figure 2. The download page of JDK

(If you encounter installation failure, try to unzip a compressed archive and configure environment variables. Refer to [English](#) or [Chinese](#) tutorials for preparing Java.)

Start up 'Mars4_5.jar'. If you cannot run it directly by double clicks, call a command line window in the directory where the jar is located. Then run the following command:

```
java -jar Mars4_5.jar
```

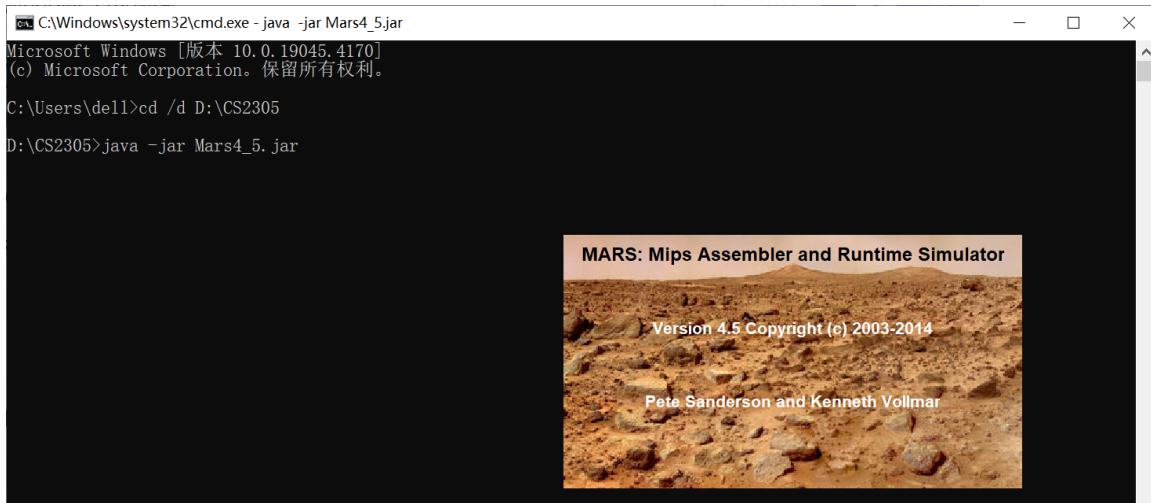


Figure 3. Run the MARS

2 Basic MARS Use (5 point)

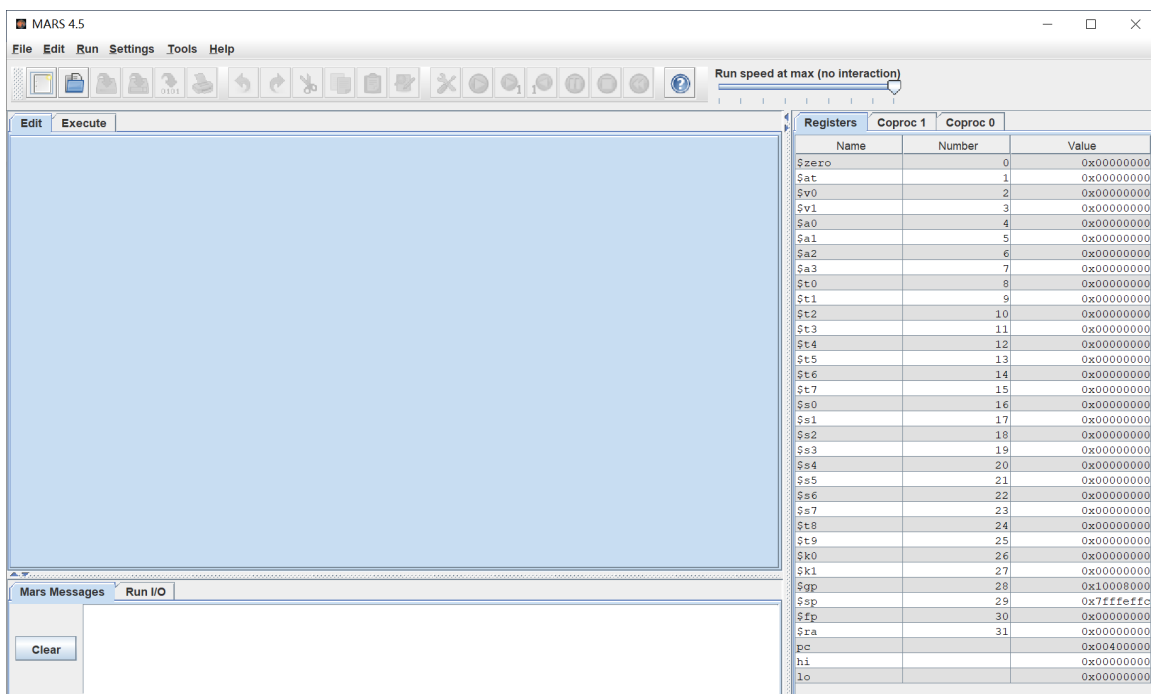


Figure 4. The interface of MARS

1. Begin creating a new MIPS assembly file by:

Click the icon  or select File -> New from the menu bar.

2. In the Edit window, you can type in MIPS assembly instructions. For example,

```

1  # file: example.s
2  .data
3  string1: .asciiz "hello! My name is leezeeyee\n"
4  .text
5  main:
6
7  li $v0, 4
8  la $a0, string1
9  syscall
10
11 li $v0, 10
12 syscall

```





Figure 5. An example of MIPS assembly program

3. Using Ctrl+S to save the program file.

4. Assemble the program using the icon



5. Choose how you will execute the program:

- The  icon runs the program to completion. Using this icon, you should observe the yellow highlight showing the program's progress and the values of the Fibonacci sequence appearing in the Data Segment display.
- The  icon resets the program and simulator to initial values. Memory contents are those specified within the program, and register contents are generally zero.
- The  icon is "single-step." Its complement is , "single-step backwards" (undoes each operation).

6. Locate the Registers display, which shows the 32 common MIPS registers. Other tabs in the Registers display show the floating-point registers (Coproc 1) and status codes (Coproc 0).

7. Observe the output of the program in the Run I/O display window:

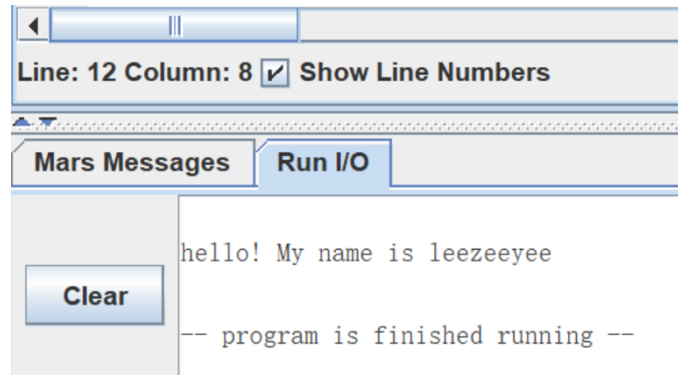


Figure 6. The output of the program

Modify the sample program to print your student ID and name, and display a screenshot of it in the report. (5 point)

3 Task: Branch

3.1 Coding (25 Points)

Write a MIPS assembly program to achieve the same functionality as the following C code. Variables *A*~*C* and *Z* must be integer in memory, which can be loaded into registers during program execution, and their initial values can be modified before each run. Note that pseudo-instructions related to branches in MIPS can be used to simplify the comparison and branching code. It is recommended to display the values of the data segment in decimal to facilitate result observation.

Please provide the screenshots of the code in your report.

```
//Branch.c
int main()
{
    //Note: I should be able to change the values of A, B, and C when testing
    //your code, and get correct output each time!(i.e.don't just hardwire your output)
    int A = 10; int B = 15; int C = 6; int Z = 0;
    if (A > B || C < 5)
        Z = 1;
    else if (A == B)
        Z = 2;
    else
        Z = 3;

    switch (Z)
    {
        case 1:
            Z = -1;
            break;
        case 2:
            Z = -2;
            break;
        default:
            Z = 0;
            break;
    }
}
```

Figure 7. The C code of branch

As a hint, you can complete the following assembly code.

```

1  # file: branch.asm
2  # author: zhangbowen
3  # date: 2024/04/08
4  .data
5  A: .word 10
6  B: .word 15
7  C: .word 6
8  Z: .word 0
9  .text
10 main:
11 lw $t5,A          # Load data A in $t5
12 lw $t6,B          # Load data B in $t6
13 lw $t7,C          # Load data C in $t7
14 lw $t8,Z          # Load data Z in $t8
15 bgt $t5,$t6,L10   #A>B
16 blt __,__,_       #C<5
17 j L110
18 L10:
19 li $t8,1          #Z=1
20 j L20
21 L110:
22 __ __,__,_        #A==B
23 li $t8,3          #Z=3#eLse
24 j L20
25 L11:
26 li $t8,2          #Z=2

```

```

27 L20:
28 beq $t8,1,L3      #z==1
29 j L4
30 L3:
31 li $t8,-1         #Z=-1
32 j L7               #break
33 L4:
34 _ $t8,_,_         #z==2
35 _ _               #default
36 L5:
37 li $t8,-2         #Z=-2
38 j L7               #break
39 L6:
40 li $t8,0          #Z=0
41 L7:
42 sw $t8,Z

```

Figure 8. The hint for branch task

3.2 Results Display (20 Points)

This task requires showcasing the results after running the program with different initial values. Please provide the screenshots of the running results in your report as shown in the following figure (the screenshot should include both the Text Segment and the Data Segment).

Value(+0), Value(+4), Value(+8), and Value(+c) represent the values of *A*, *B*, *C*, and *Z*, respectively.

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	10	15	3	-1	0

Figure 9. The result display of branch task

Try the following four test cases:

Case 1

Input: A=10, B=15, C=1

Output: Z=-1

Case 2

Input: A=15, B=10, C=6

Output: Z=-1

Case 3

Input: A=15, B=15, C=6
Output: Z=-2

Case 4

Input: A=10, B=15, C=6
Output: Z=0

3.4 In-depth Investigation (10 Points)

Question: Continue with the given assembly code. Line 15 shows an instruction '**bgt \$t5, \$t6, L10**'. Given A=15 and B=10, which components get involved in this instruction?

- a. Program Counter (PC)
- b. Instruction Memory
- c. Registers (write port)
- d. Registers (read port)
- e. Data Memory (write port)
- f. Data Memory (read port)
- g. ALU
- h. PC+4 Adder
- i. Branch Adder

Tips: You can use the tool 'MIPS X-Ray' to get an in-depth investigation.

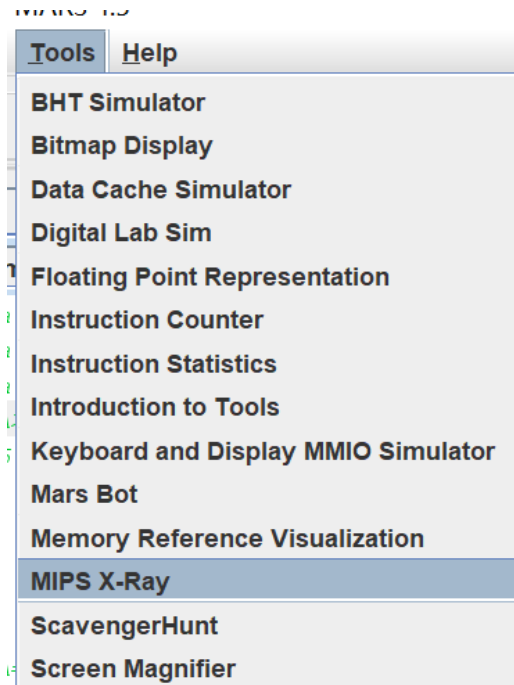


Figure 10. Open the MIPS X-Ray

Click 'Connect to MIPS' and the icon 'compile'.

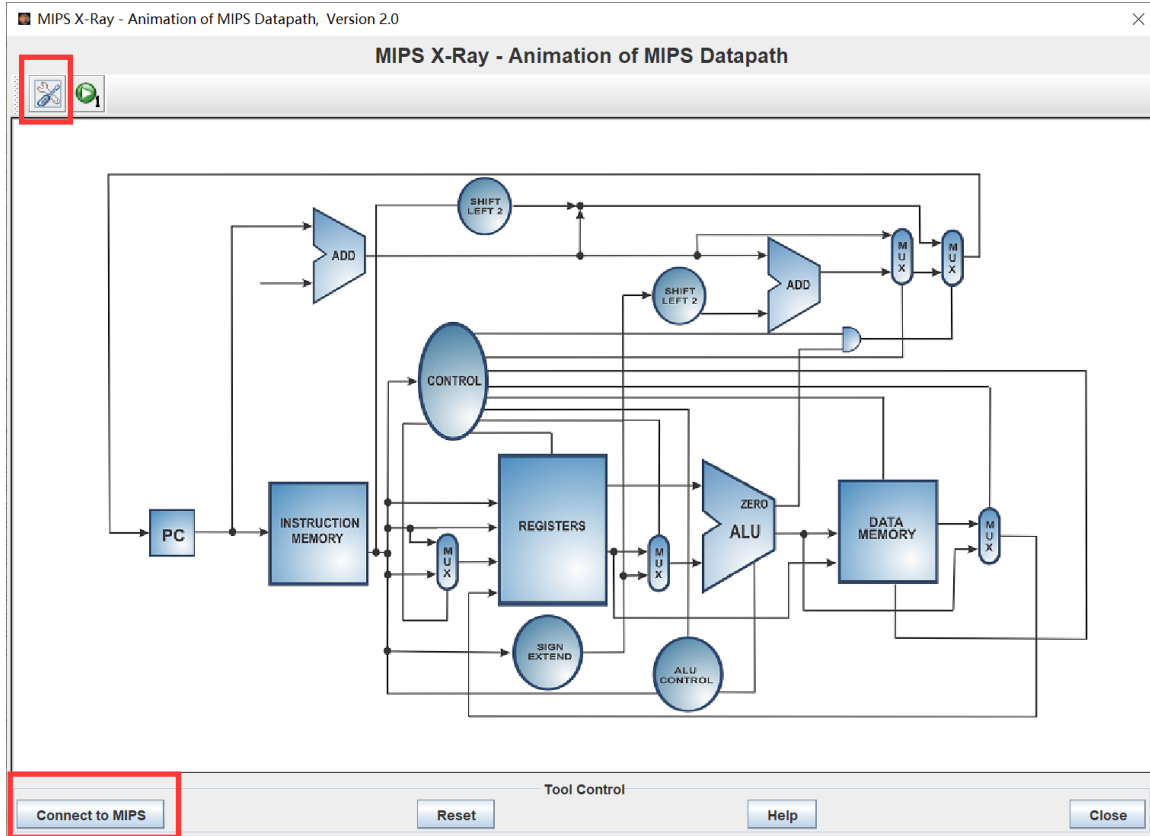


Figure 11. The interface of MIPS X-Ray

Step one by one until the target instruction. The below figure (right-hand) illustrates the involved components in the datapath for "slt" instruction.

s Help

MIPS X-Ray - Animation of MIPS Datapath, \

Basic

01 lui \$1,0x00001001	11: lw \$t5,A# load data A in
00 lw \$13,0x00000000(\$1)	
01 lui \$1,0x00001001	12: lw \$t6,B# load data B in
04 lw \$14,0x00000004(\$1)	
01 lui \$1,0x00001001	13: lw \$t7,C# load data C in
08 lw \$15,0x00000008(\$1)	
01 lui \$1,0x00001001	14: lw \$t8,Z# load data Z in
0c lw \$24,0x0000000c(\$1)	
0e slt \$1,\$14,\$13	15: bgt \$t5,\$t6,L10#A>B
03 bne \$1,\$0,0x00000003	
05 slti \$1,\$15,0x000000...	16: blt \$t7,5,L10#C<5
01 bne \$1,\$0,0x00000001	
0f j 0x0040003c	17: j L110
01 addiu \$24,\$0,0x0000...	19: li \$t8,1#Z=1

Value (+0)	Value (+4)	Value (+8)
0x0000000a	0x0000000f	0x00000006
0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000

0x10010000 (.data)

Control Signals
Active Inactive

Instruction
slt \$t6 at \$t5

opcode rs rt rd sh
000000 01110 01101 00001 0

Disconnect from MIPS

Figure 12. The usage of MIPS X-Ray

4 Task: Loop

4.1 Coding (10 Points)

Write a MIPS assembly program to achieve the same functionality as the following C code. Variables *i* and *Z* must be integer variables in memory, which can be loaded into registers during program execution, and their initial values can be modified before each run. Note that pseudo-instructions related to branches in MIPS can be used to simplify the comparison and branching code. It is recommended to display the values of the data segment in decimal to facilitate result observation.

Please provide the screenshots of the code in your report.

```
// Loop.c
int main()
{
    int z = 100;
    int i = 0;
    do
    {
        i++;
    } while (i < z);

    while (i > 0)
    {
        z += i;
        i--;
    }
}
```

Figure 13. The C code of loop

As a hint, you can complete the following assembly code.

```

# file: loop.asm
# author: bowenzhang
# date: 2024/4/8
.data
Z: .word 100
i: .word 0
.text
main:
lw $t1,Z           # load data Z in $t1
lw $t2,i           # load data i in $t2
Loop1:
add __,__,1        # i++
blt __,__,Loop1    # i<Z

bgt __,__,Loop2    # i>0
Loop2:
__ --'--'--
__ --'--'--
__ --'--'--

# store result
sw $t1,Z
sw $t2,i

```

Figure 14. The hint for the loop task

4.2 Result Display (10 Points)

This task requires showcasing the results after running the program with different initial values. Please provide the screenshots of the running results in your report as shown in the following figure (the screenshot should include both the Text Segment and the Data Segment).

Value(+0), Value(+4) represent the values of z and i , respectively.

Address	Value (+0)	Value (+4)
0x10010000	60	0

Figure 15. The result display of loop task

We have the following two test cases:

(1) $Z=100, i=0$

(2) $Z=99, i=0$

4.3 Optimization (20 Points)

Optimize the Loop2 in the above assembly code using loop unrolling to ensure that the total number of instructions required to execute two test cases does not surpass 800. The method to query the number of instructions required for program execution is as follows:

1. Select Tools->Instruction Counter

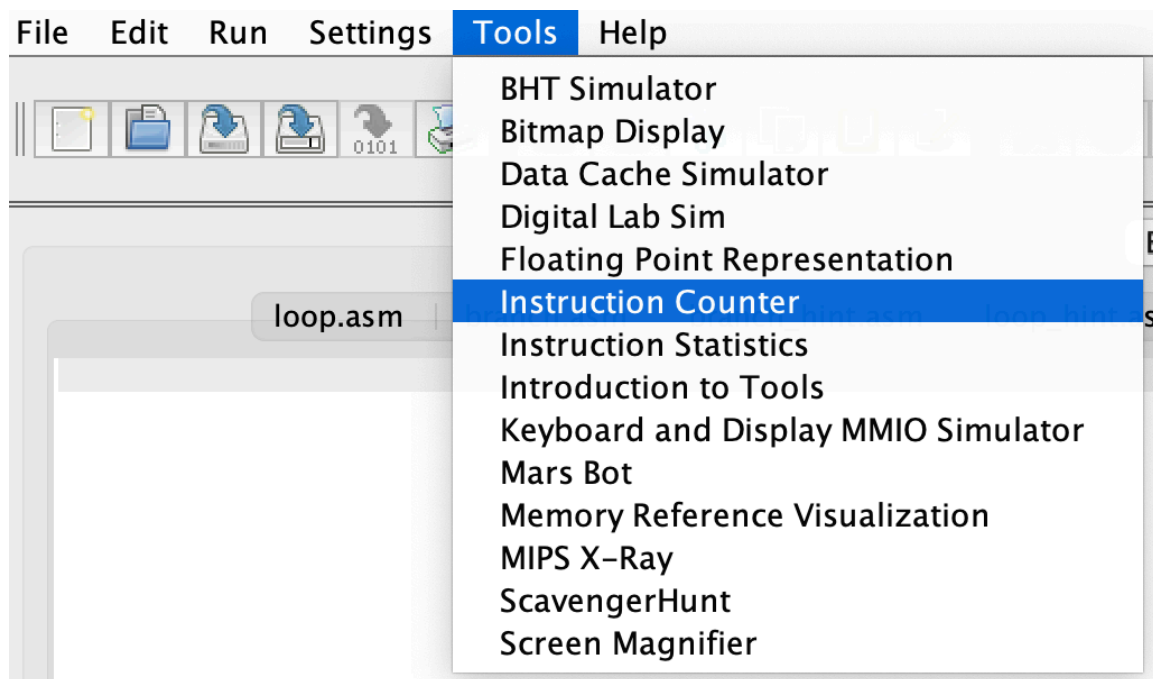


Figure 16. The usage of instruction counter (1)

2. Click “Connect to MIPS”

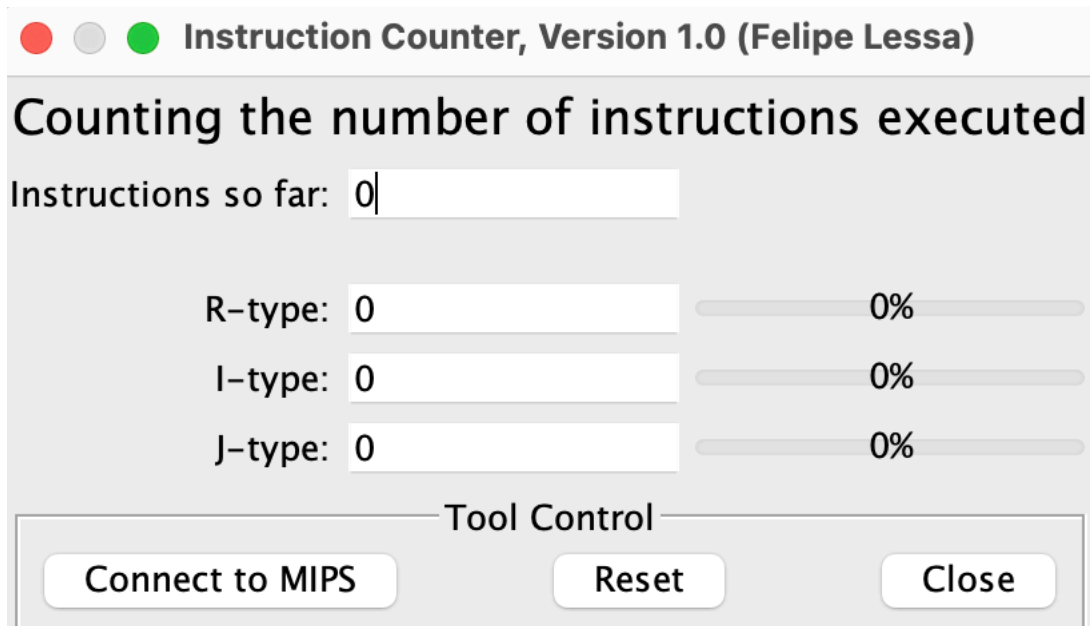


Figure 17. The usage of instruction counter (2)

3. Compile and run your code

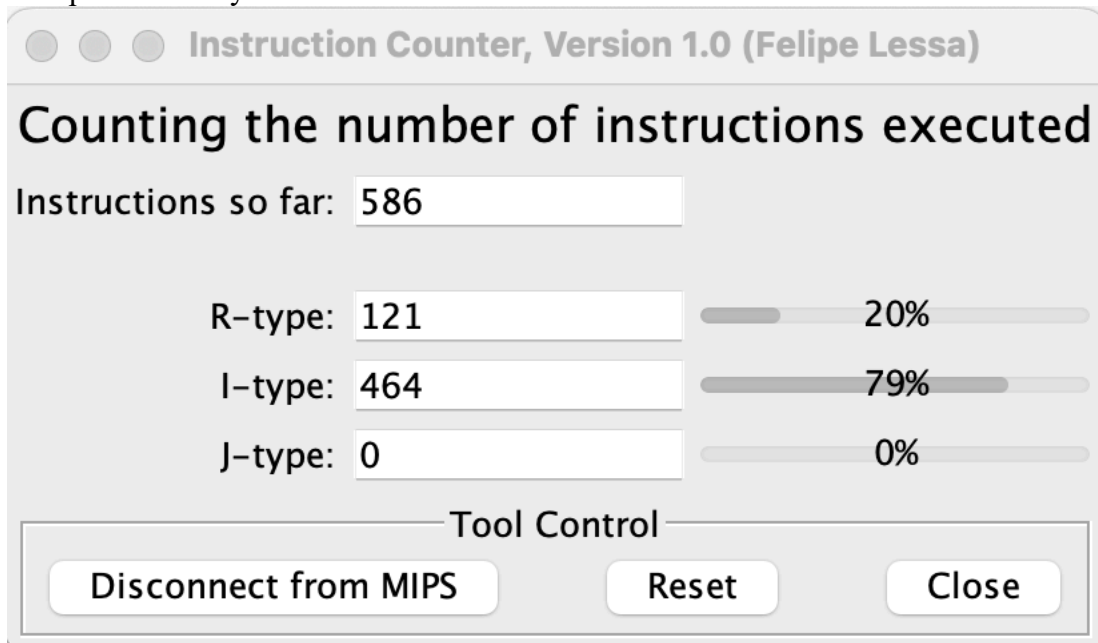


Figure 18. The usage of instruction counter (3)

Please elaborate on the optimization methods used in the report and provide a screenshot showing the number of instructions required for program execution.

5 Submission Formatting

Submit a compressed file named "studentID+Name.zip" (e.g., 517021910797+bowenzhang.zip), which includes branch.asm, loop.asm, loop_unrolling.asm, and a report named "studentID+Name.pdf". The report must include answers to the requirements in highlighted sections.