



MIDDLE EAST TECHNICAL UNIVERSITY
NORTHERN CYPRUS CAMPUS

CNG 331 – COMPUTER ORGANIZATION
2019-2020 FALL TERM PROJECT
MIPS SIMULATOR

Günseli Sevinç – 2315521

İdil Ece Trabzon - 2152189

INTRODUCTION TO PROBLEM

In this term project, we were supposed to design a mips simulator which takes the instruction as an input and converts it to the hexadecimal. Firstly, we were going to take the instruction, separate them according to their opcodes, registers, shamts, functions, immediate numbers and addresses by first deciding the type of the instruction. After separation, we needed to assign them according to their values by using mips reference data. Finally, we need to create an array, put the binary numbers to that array which is 32 size and convert that numbers into hexadecimal.

Our Mips Simulator can work in two different modes. These are Interactive Mode and Batch Mode. Interactive Mode works on the command line, user enters how many instructions s/he wants to enter and the program gets that many instructions to convert hexadecimal. And batch mode reads the instructions from a file and writes the hexadecimal outputs to a file as well.

ASSEMBLER DESIGN CHOICE

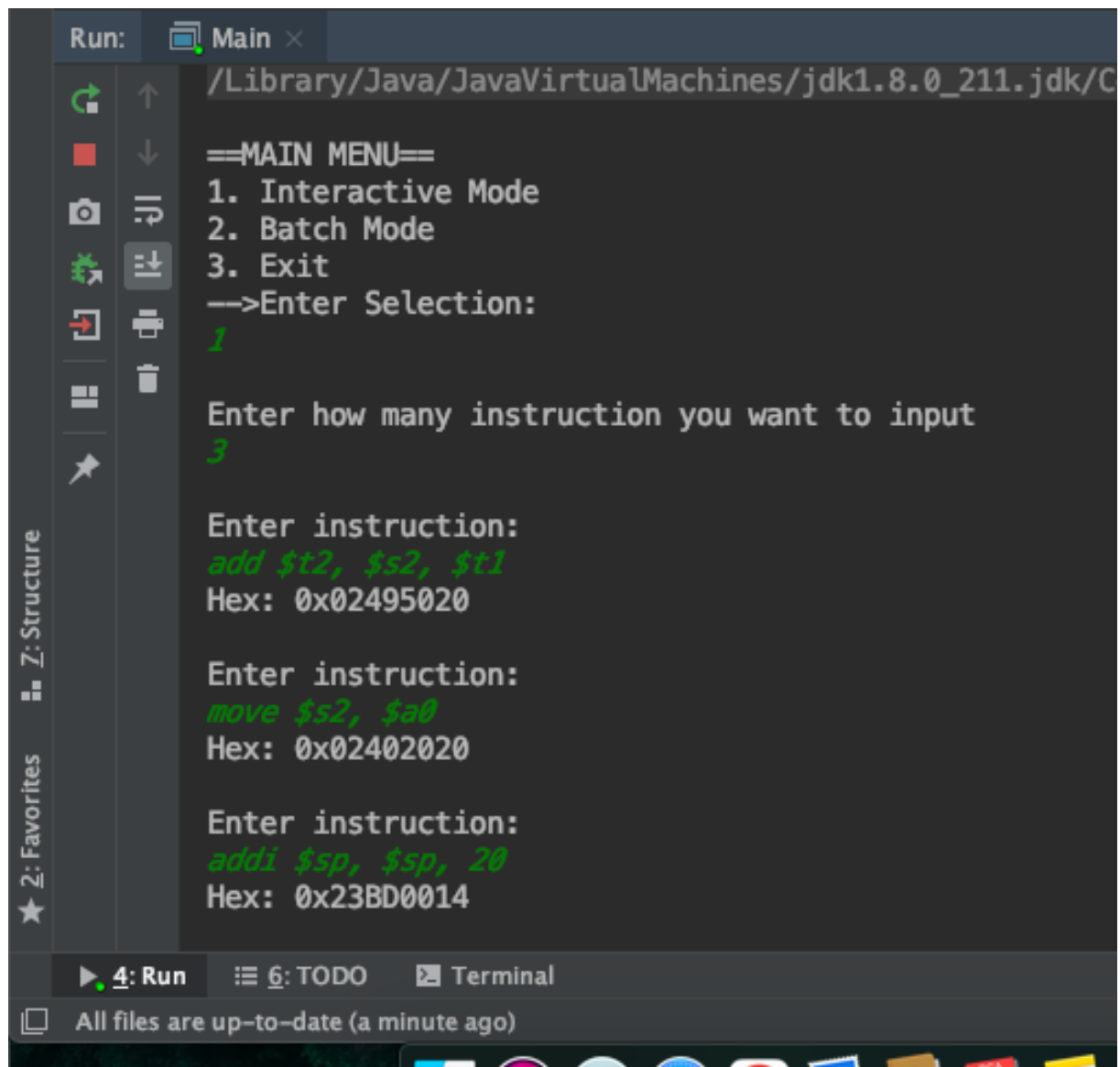
To do this project, we chose the Java language. Because of Java has a lot of methods itself, we think that it will be easy to split the instructions (by split method), putting their binary numbers (by using hash tables), converting the integers to binary and hexadecimal numbers and converting the integers to strings. In addition, using hash tables gave us so much benefit while writing the instruction in binary. We took formats of the instructions like opcodes, rs, rt from these tables.

CONCLUSION

To conclude that we learned the principles of the MIPS converter. We learned how to convert a mips code to the hexadecimal code. Firstly, it should decide the type of the instruction and then according to the given input, it should place the true binary numbers to the mips array, and then compute and print the hexadecimal code. We used hash tables for flexibility. It gives us a chance of writing all the rs, rt, rd , immediate , opcode, shampt, function and address from the hash tables. We also experienced reading instructions from a file and writing the conversions also another one. Unfortunately, we couldn't achieve some instructions like j type instructions and beq, bne because we couldn't handle the Label or exit part. Using java, gave us a lot of of benefits like we could compute, binary to integer, integer to string or binary to hexadecimal conversions. We compute the add, move, sll, slti, lw, sw, addi and jr instructions properly. In addition we used twos complement for negative numbers in addi instruction. To sum up, we had experience on how the mips converter works.

How It Works - Interactive Mode

add, move, addi Instructions



```
Run: Main x
/Library/Java/JavaVirtualMachines/jdk1.8.0_211.jdk/C

==MAIN MENU==
1. Interactive Mode
2. Batch Mode
3. Exit
-->Enter Selection:
1

Enter how many instruction you want to input
3

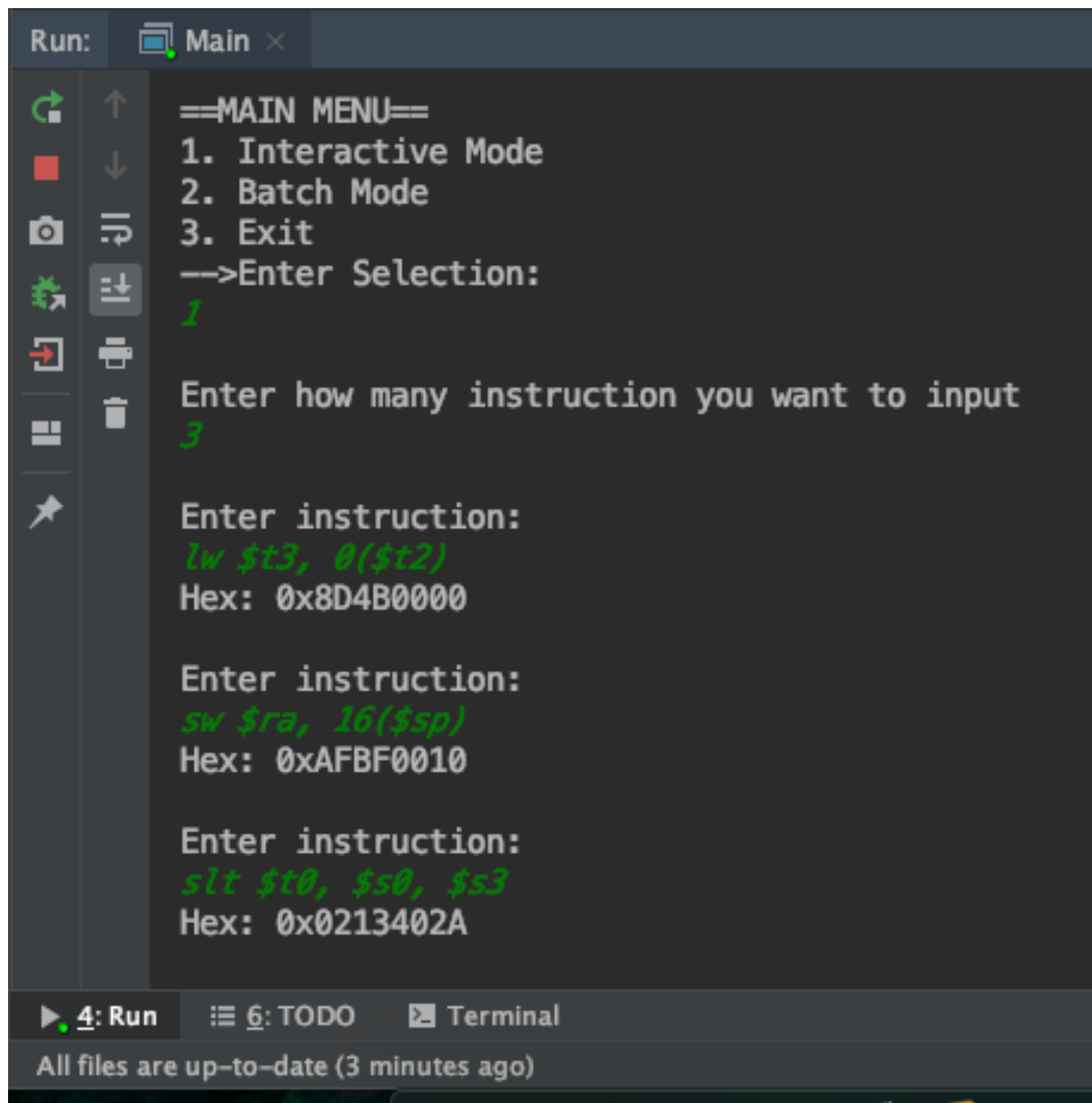
Enter instruction:
add $t2, $s2, $t1
Hex: 0x02495020

Enter instruction:
move $s2, $a0
Hex: 0x02402020

Enter instruction:
addi $sp, $sp, 20
Hex: 0x23BD0014

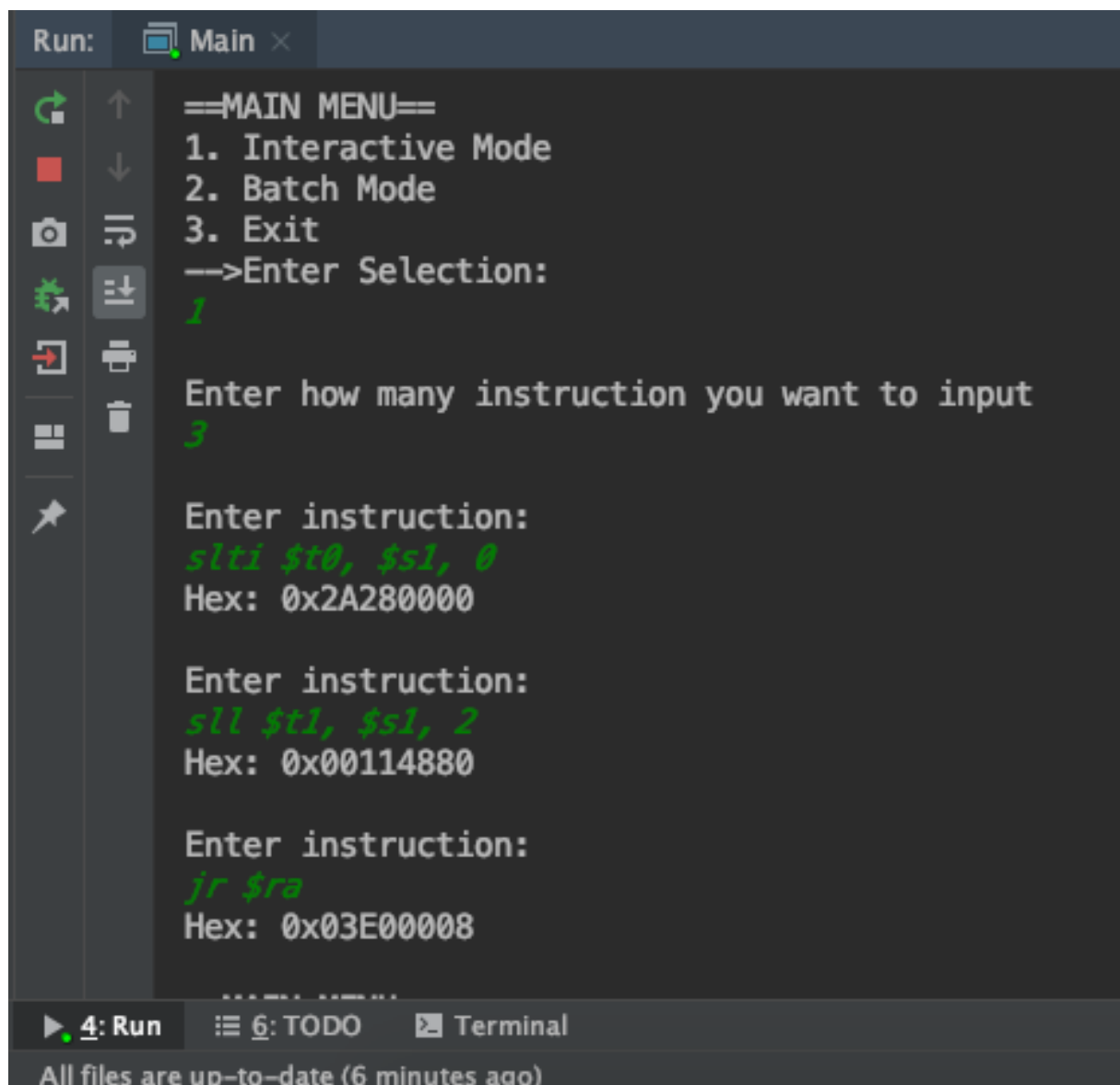
4: Run 6: TODO Terminal
All files are up-to-date (a minute ago)
```

lw, sw, slt Instructions



```
Run: Main x
==MAIN MENU==
1. Interactive Mode
2. Batch Mode
3. Exit
-->Enter Selection:
1
Enter how many instruction you want to input
3
Enter instruction:
lw $t3, 0($t2)
Hex: 0x8D4B0000
Enter instruction:
sw $ra, 16($sp)
Hex: 0xAFBF0010
Enter instruction:
slt $t0, $s0, $s3
Hex: 0x0213402A
4: Run 6: TODO Terminal
All files are up-to-date (3 minutes ago)
```

slti, sll, jr Instructions

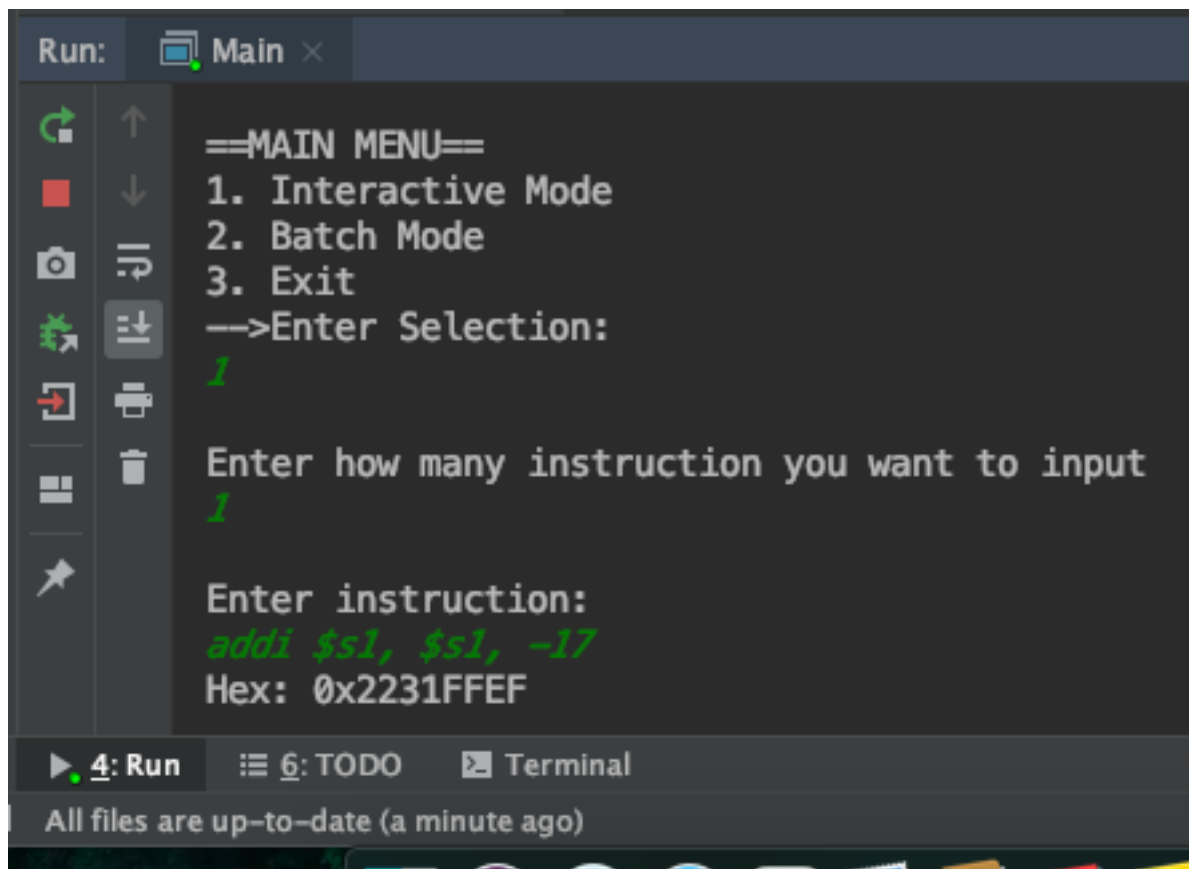


The screenshot shows a code editor window titled "Main" with a dark theme. On the left is a vertical toolbar with icons for running, undo, redo, and other development actions. The main area contains the following text:

```
==MAIN MENU==  
1. Interactive Mode  
2. Batch Mode  
3. Exit  
-->Enter Selection:  
1  
  
Enter how many instruction you want to input  
3  
  
Enter instruction:  
slti $t0, $s1, 0  
Hex: 0x2A280000  
  
Enter instruction:  
sll $t1, $s1, 2  
Hex: 0x00114880  
  
Enter instruction:  
jr $ra  
Hex: 0x03E00008
```

At the bottom, there is a status bar with tabs for "4: Run", "6: TODO", and "Terminal". Below the tabs, it says "All files are up-to-date (6 minutes ago)".

addi \$s1, \$s1, -17



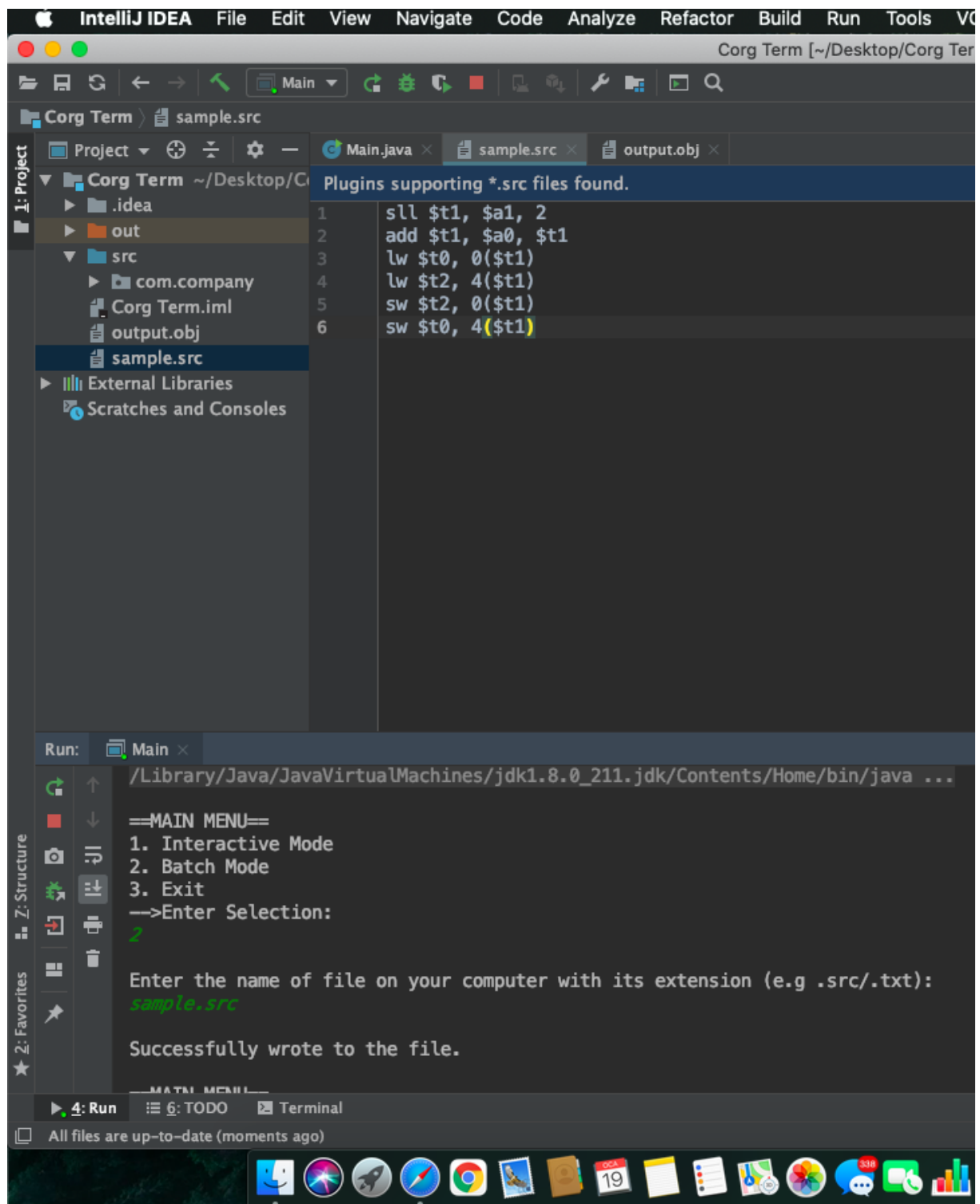
The screenshot shows a terminal window with a dark background. The title bar at the top says "Run: Main x". On the left side, there is a vertical toolbar with icons for running, stepping through code, and other development actions. The main area of the terminal displays the following text:

```
==MAIN MENU==  
1. Interactive Mode  
2. Batch Mode  
3. Exit  
-->Enter Selection:  
1  
  
Enter how many instruction you want to input  
1  
  
Enter instruction:  
addi $s1, $s1, -17  
Hex: 0x2231FFEf
```

At the bottom of the terminal window, there is a status bar with tabs for "4: Run", "6: TODO", and "Terminal". Below the tabs, it says "All files are up-to-date (a minute ago)".

How It Works - Batch Mode

Sample.src File



output.obj File

