CSCI 6461 Fall 2023 Sec 10 Group 10 Project III – Hand Guide

JDK Compiler compliance level : 17
JRE version : jre.full.win32.x86_64_17.0.8.v20230831-1047
Refer to "CSCI 6461 Fall 2023 Sec 10 Group 10 Project 1 – Hand Guide" and "CSCI 6461 Fall 2023 Sec 10 Group 10 Project IIa – Hand Guide" to know more about our console.

Project III :

The project objective is to execute the instructions that were part of the previous project, project IIa. In this project, exceptions were modified an we have added new methods to handle each instruction accordingly. Cache is implemented using HashMap data structure available in java. Follow Simulator.java about cache implementation.

Changes in console :

Introduced CC bit as part of this project. CC bit has 4 bits. CC[0] is set when there is OVERFLOW, CC[1] is set when there is UNDERFLOW, CC[2] is set when we encounter DIVIDE BY ZERO in case of DVD instruction and CC[3] is set when TRR instruction yields true(verifies if c(rx) == c(ry)).

To implement IN and OUT instructions, two new text areas are introduced in the console. One is editable for keyboard input and the other is non editable, for printer output.

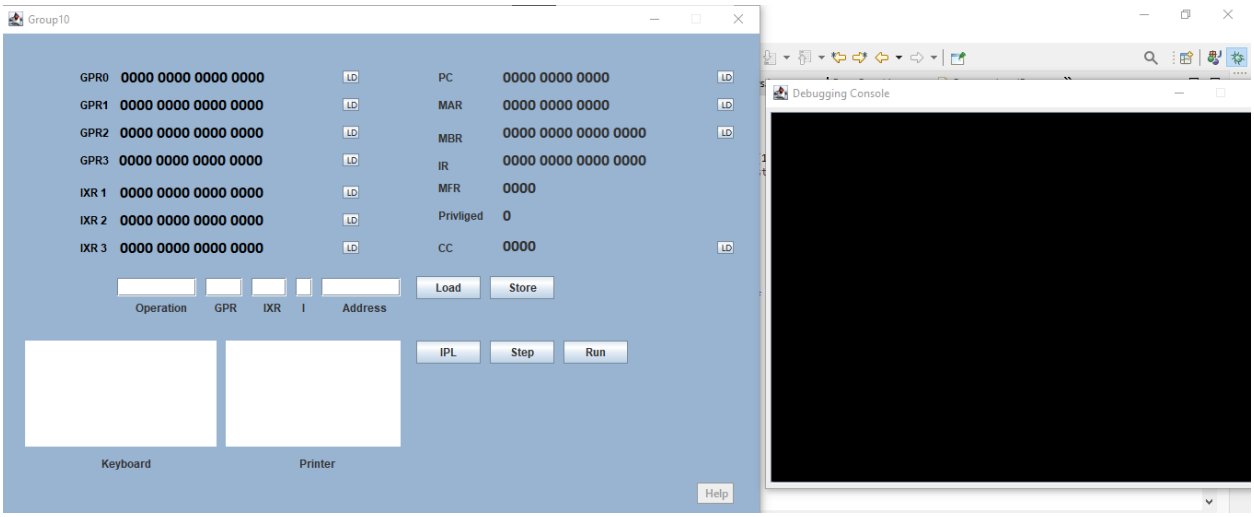
Starting the Console:

Note : Make sure that **ProgramLoadFile.txt**, **HexFile.txt, InstructionsMetaData.json** and the jar file are in the same level in the folder structure.

- Launch the console by running the command

java -jar CSCI6461F23S10G10Project3.jar

- You shall see the input below. One window is the console and the other is the debugging console, to see what is happening in the console.
- In ProgramLoadFile.txt, actual instructions start from location 18. So, enter the value of PC as **000000 00 00 0 10010** and click **LD** button that corresponds to PC register.
- Use the value **000000 00 00 0 10101** to set **GPR3**. Click on **LD** button that corresponds to GPR3
- Click the **IPL** button. Clicking IPL button is mandatory to load the HexFile.txt contents into the memory. Else the program doesn't run. You need to restart the program incase if you unknowingly clicked "Step" button without clicking IPL.



- At the start of the program instructions, we have IN instruction. So, it asks for user input. You can give any character, but only one, or any number(positive integers) in the keyboard area (if you want to RUN the console) or else you can also give input when you encounter an exception(if you want to execute step by step).
- Now click on the step button repeatedly to see the different registers change their values for each step. Consequently, the register which was updated can be seen in the debugging console.
- At the end of the program, you shall see a window saying “HALT triggered. Program Halted. End of the program.”. It means the program was executed completely and reached the end.

*If you encounter any error or exception, try to correct the program and restart the program by running the command above.

ProgramLoadFile.txt - Description

```
LOC 1               -start at location 1
 Data 17            -store 17 at memory location 1
 Data 1
 Data 6
 Data 14
 Data 50
 Data 12
 Data 9
 Data 39
 Data 27
 Data 10
 Data 7
 Data 80
 Data 90
 Data 4
 Data 60
```

```
 Data 31
 Data 3
 IN 2,0             -Take input from device id 2 and store in GPR0
 RFS 21             -Return from subroutine
 LDX 1,7            -Load IXR1 with EA(7)
 SRC 0,2,1,1        -Logically shift GPR0 for a count of 2 to the left
 OUT 2,1            -output the value of GPR2 to the device id = 1
 JNE 0,0,9          -Jump On Not equal to Zero when GPR0 value is zero
 SRC 3,3,1,1
 LOC 27             -Change location to 27
 SOB 0,1,8          -Subtract and branch
 SRC 1,4,0,1        -Logically right shift GPR1 for count of 4
 LDX 3,1
 LOC 39             -Change location to 39
 LDX 2,6
 LDR 2,0,3
 LDR 3,0,4
 OUT 0,1
 ORR 2,3            -perform OR operation of c(gpr2) and c(gpr3) and store in gpr2
 RRC 2,2,0,1        -Rotate logically GPR2 for a count of 2 to the right
 AND 3,2            -Perform AND operation of c(gpr3) and c(gpr2) and store in gpr3
 NOT 2              -perform NOT on gpr2
 JZ 2,0,50          -Jump on zero if c(gpr2) is zero
 SRC 1,3,0,0
 LOC 50             -change location to 50
 TRR 2,3            -perform equality operation on c(gpr2) and c(gpr3). If equal set
                    cc(3) bit
 LDR 0,0,1          - Load register 1
 LDR 2,0,7
 MLT 2,0            -Multiply c(gpr2) with c(gpr0) and store result in gpr2 and gpr3
 SRC 0,14,0,1
 JGE 2,0,15         -Jump if greater than or equal to zero on c(gpr2)
 LOC 60             -Change location to 60
 LDR 2,0,2
 LDR 0,0,1
 DVD 0,2            -Divide c(gpr0) and c(gpr2) and store result in gpr0 and gpr1. If
                    there is divided by zero, set cc(2) bit
 SIR 2,5            -subtract c(gpr2) with 5
 SRC 2,2,0,0
 OUT 2,1
 LDX 1,6
 JMA 1,2            -Unconditional jump to addr(2)
 AMR 1,0,3          -Add to memory register gpr1
 SMR 1,0,4          -Subtract from memory register gpr1
 JSR 3,12           -Jump and save return address
 LOC 80             -change location to 80
 AIR 2,10           -Add 10 to c(gpr2)
 SIR 2,3            -Subtract 3 from c(gpr2)
 JCC 3,0,14         -Jump on conditional code
 LDX 1,2
 STR 2,0,1,1        -store c(gpr2) into memory
 LOC 90             -change location to 90
 STR 1,0,4
 LDR 0,0,1
End: HLT            -End of the program
```