This simulator simulates the microarchitecture Mic-1 as shown on page 242 of the textbook. The simulator shows contents of all ten registers, MIR, MPC, ALU's inputs & output, and the console (standard out).

The behavior of the simulator is controlled by a microprogram that is stored in the control store (512 words of 36 bits each). The microprogram is a sequence of microinstructions where each microinstruction is a set of control signals that drives the datapath for one CPU cycle. The micro program is given as a binary file (*.mic1 - cannot be displayed or printed) that can be understood by the simulator.

The microprogram interprets (fetches, analyzes and executes instruction by instruction) a Level-2 program (macro program which is in integer java virtual machine bytecode). This macro program is given as a binary file (*.ijvm – cannot be displayed or printed).

To minimize the complexity of creating a binary macro program, a high level assembly language (JAS – Java Assembly Language) is used to create a macro program. All instructions that are supported by this JAS are given in ijvm.conf (IJVM configuration) ascii file. An IJVM Assembler (ijvmasm in L2 subdirectory) is provided that can translate a high-level macroprogram (*.jas) according to IJVM configuration (ijvm.conf) and create a binary version of the macroprogram (*.ijvm).
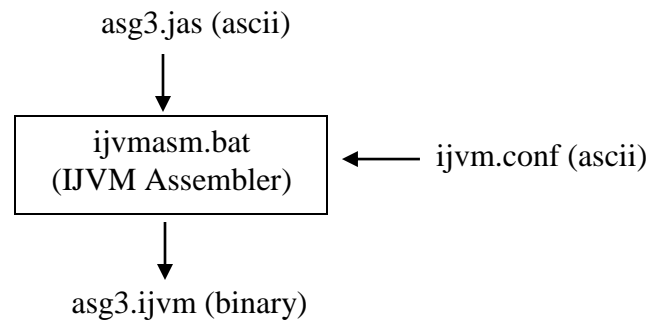
Creating a binary microprogram is very cumbersome. To simplify this task, a microprogram can be written in a micro assembly language (*.mal – ascii file) format as depicted in Figure 4-16 (page 258) and Figure 4-17 (page 262). Note that all memory operation (fetch, read, or write) takes two cycles (i.e. if a memory fetch operation is initiated in 7th cycle, the value can be used only in 9th or later cycles). A micro assembler (mic1asm in L1 subdirectory) has been included in this package that compiles a micro assembly language program (*.mal) into a binary version microprogram (*.mic1).

Once we have a microprogram (binary version) and a macroprogram (binary version), the simulator (mic1sim in Simulator subdirectory) can be invoked. The File menu allows us to interactively select the microprogram as well as macroprogram. The Step button executes one microinstruction and waits for next user response. The Reset button will reset the execution. The Run button initiates the execution until completion. The Stop button (in case of infinite loop) terminates execution.
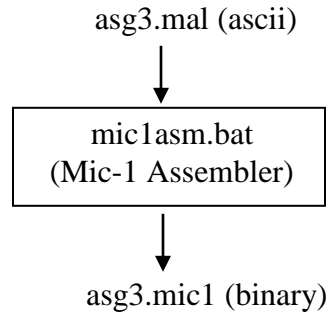
For asg3, the macroprogram asg3.jas and the corresponding asg3.ijvm along with ijvm.conf will be given. Also, stdmic1.mal (microprogram that interprets the standard 24 IJVM instructions) is provided. The objective of asg3 is to extend stdmic1.mal to interpret additional three IJVM instructions (IMULT, IDIVI, and OUTBIN) and prepare asg3.mal. After compiling asg3.mal into asg3.mic1 (using mic1asm), run the simulator with asg3.mic1 (microprogram) to interpret asg3.ijvm (macroprogram). Good Luck!
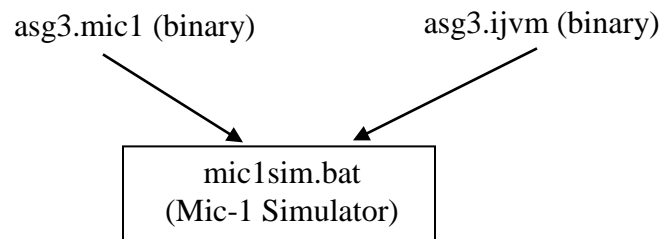
--Prabakar

Creating a macroprogram:

asg3.jas (ascii)

↓

```
ijvmasm.bat
(IJVM Assembler)
```
← ijvm.conf (ascii)

↓

asg3.ijvm (binary)

Creating a microprogram:

asg3.mal (ascii)

↓

```
mic1asm.bat
(Mic-1 Assembler)
```

↓

asg3.mic1 (binary)

Running the simulator:

asg3.mic1 (binary)   asg3.ijvm (binary)

```
mic1sim.bat
(Mic-1 Simulator)
```

ASG3 Startup steps

- Download Asg3 Zip file (Project3.zip) and uncompress it

- Read Readme.pdf file

- Edit env.bat files (change JDK path to your current JDK bin folder) in L1\, L2\, and Simulator\ directories

- Run simulator & test with OUTBIN\OutBinNibble.mic1 and OUTBIN\OutBin.ijvm

- Read L1\MALtips.txt

- Print OUTBIN\OutBinNibble.pdf

- Edit asg3\asg3.mal and append OUTBIN segment (44 micro instructions)

- Compile asg3\asg3.mal using L1\mic1asm

- Run simulator & test with asg3\asg3.mic1 and asg3\asg3.ijvm for OutBin operation

- Run simulator & test with asg3\prabu3.mic1 and asg3\asg3.ijvm for OutBin/Mult/Divide operations

- Understand asg3\asg3alg.txt - logic for Multiply and Divide