Naman Gangwani
nkg160030@utdallas.edu

CS 3340.002
December 11, 2017

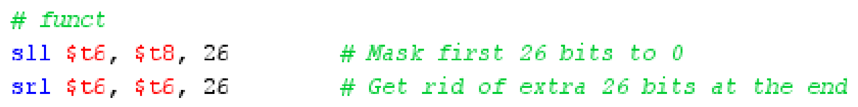
Project Report

Problem Statement

The purpose of this assignment is to read a machine code file from a file provided by the user and disassemble it to ultimately output in MIPS assembly syntax. This is important in creating a better understanding of how programs are assembled, because knowledge of disassembling can persist with the knowledge of assembling. As for the program, the file does not necessarily have to have a new line at the end, but it must be readable by a scanner, and each line must have 32 bits to represent an instruction. The output should display the properly disassembled instructions of machine code from all the information given in *Appendix A* and the *Opcodes, Base Conversion, ASCII Symbols* table from the book Computer Organization and Design, the Hardware-Software Interface by David A. Patterson and John L. Hennessy, excluding pseudoinstructions. Furthermore, the supplied file “fibs.txt” may be used as a sample file, but the program must work for all valid files that meet the criteria.

Approach to Solution

The printing of disassembled instructions in the specified file is done by utilizing the MIPS assembly language with the Mars IDE, using tools that assemble and run the program on a computer that runs the Java Virtual Machine (JVM). Knowledge of the working with the basic structure of loops, various system calls for performing specific actions, functional programming, file reading, retrieving string lengths, evaluating ASCII codes, converting from binary to decimal, and shifting logically and arithmetically are essential. It is important to understand that many different modes of instructions can be printed for different types, and from *Appendix A* in the book referenced in the problem statement, one can derive that there are 8 different ways to print R-types, 5 for I-types, 1 for J-types, 4 for FR-types, and 1 for FI-types. One pivotal point to note is how to retrieve specific fields from a line of machine code, which can be done with shifting logically or arithmetically accordingly to the field being read (as shown in the screencap below).

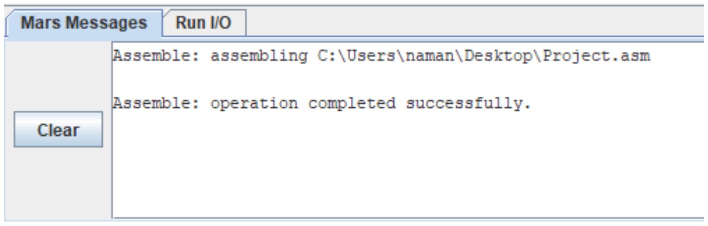


```
# funct
sll $t6, $t8, 26        # Mask first 26 bits to 0
srl $t6, $t6, 26        # Get rid of extra 26 bits at the end
```

Retrieving the function code of an R-type consists of logically shifting the numerical value of the line of binary text from the file to the left by 26 bits and then logically back to the right 26 bits again.

Solution Description

When writing the program, old code such as opening a file from a user inputted filename, retrieving the length of strings, printing strings and integers, line processing, and converting from binary to decimal have been reused, but handling specific output cases for different instruction types and retrieving corresponding values from a table of preset strings have been written from scratch. When the Mars IDE has been launched and Project.asm has been opened, it is important to build (or "assemble") the program before running it. To do so, click the wrench and screwdriver button found in the tool icon layout. After clicking it, the program should successfully assemble, and there should be a success message from the "Mars Messages" tab at the bottom of the screen.

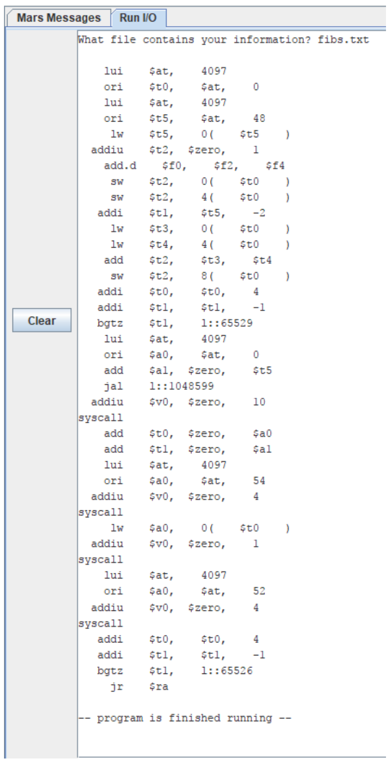


```
Assemble: assembling C:\Users\naman\Desktop\Project.asm

Assemble: operation completed successfully.
```

The message given after the Project.asm has been successfully assembled with no errors.

Once the program has successfully compiled, it is now time to run it. At the top of the screen, click the green play button in the tools to the right of the assemble button pressed earlier. The user will be prompted to enter the name of a file in the "Run I/O" tab. If the filename is valid and found in the same directory as the executable MARS launcher, the "Run I/O" tab should output instruction for each 32-bit machine code instruction as they would appear in MIPS assembly syntax, each on a separate line as shown in the screencap below. If the filename is invalid, the program will simply output a single new line and terminate. Note that the size of the file & filename are limited by the predetermined space for them in the *.data* section of the program and that label addresses are marked by *l::* with their literal integer values read from the machine code being outputted. Instructions not found in the table from the book are marked as invalid.



```
What file contains your information? fibs.txt

    lui    $at,    4097
    ori    $t0,    $at,    0
    lui    $at,    4097
    ori    $t5,    $at,    48
     lw    $t5,    0(    $t5    )
  addiu    $t2,  $zero,    1
    add.d    $f0,    $f2,    $f4
     sw    $t2,    0(    $t0    )
     sw    $t2,    4(    $t0    )
   addi    $t1,    $t5,    -2
     lw    $t3,    0(    $t0    )
     lw    $t4,    4(    $t0    )
    add    $t2,    $t3,    $t4
     sw    $t2,    8(    $t0    )
   addi    $t0,    $t0,    4
   addi    $t1,    $t1,    -1
   bgtz    $t1,    l::65529
    lui    $at,    4097
    ori    $a0,    $at,    0
    add    $a1,  $zero,    $t5
    jal    l::1048599
  addiu    $v0,  $zero,    10
syscall
    add    $t0,  $zero,    $a0
    add    $t1,  $zero,    $a1
    lui    $at,    4097
    ori    $a0,    $at,    54
  addiu    $v0,  $zero,    4
syscall
     lw    $a0,    0(    $t0    )
  addiu    $v0,  $zero,    1
syscall
    lui    $at,    4097
    ori    $a0,    $at,    52
  addiu    $v0,  $zero,    4
syscall
   addi    $t0,    $t0,    4
   addi    $t1,    $t1,    -1
   bgtz    $t1,    l::65526
     jr    $ra

-- program is finished running --
```

The machine code instructions disassembled from "fibs.txt" into MIPS assembly syntax and outputted in the Run I/O tab.