1 Project 1: MIPS Assembly Code Programming Using MARS Tool – Run Tutorial

10 points

Project 1: MIPS Assembly Code Programming Using MARS Tool – Run Tutorial

This project will introduce you to the MARS (MIPS Assembler and Runtime Simulator) tool, which is used to develop and run MIPS assembly language programs. This is a free Java applet that will run on your PC.

1. Download following items to your computer:
    a. MAR program
    b. Tutorial
    c. Sample MIPS program: **Fibonacci.asm**
2. NOTE: to run the Mars4_5.jar file on your PC, you may need to first install *Java Runtime Environment (JRE).* If so, JRE may be downloaded from the link https://www.java.com/en/download/
3. Go through Part 1 of the tutorial to learn how to use the MARS program, including running the sample MIPS programs as directed.
    a. Examine the code and step through it so that you understand the instructions used in the code.
    b. Also examine the register and data segment contents as they update during execution.
4. Run the sample MIPS program **Fibonacci.asm.**
    a. **Take a screen shot of the MARS console (including Run/IO section) so that the grader can view your results.** If you take a screen shot of the entire console, please crop and enlarge the Run/IO section so that it is readable.
5. Turn in your code and output (screen shot) for credit. **Make sure that your name appears on <u>both</u> documents that you submit so that you can get proper credit for your work.**

2 SOLUTION

2.1 Code: Fibonacci.asm

```
# Compute first twelve Fibonacci numbers and put in array, then print
      .data
fibs: .word   0 : 12        # "array" of 12 words to contain fib values
size: .word  12             # size of "array"
      .text
      la   $t0, fibs        # load address of array
      la   $t5, size        # load address of size variable
      lw   $t5, 0($t5)      # load array size
      li   $t2, 1           # 1 is first and second Fib. number
      add.d $f0, $f2, $f4
      sw   $t2, 0($t0)      # F[0] = 1
      sw   $t2, 4($t0)      # F[1] = F[0] = 1
      addi $t1, $t5, -2     # Counter for loop, will execute (size-2)
times
loop: lw   $t3, 0($t0)      # Get value from array F[n]
      lw   $t4, 4($t0)      # Get value from array F[n+1]
      add  $t2, $t3, $t4    # $t2 = F[n] + F[n+1]
      sw   $t2, 8($t0)      # Store F[n+2] = F[n] + F[n+1] in array
      addi $t0, $t0, 4      # increment address of Fib. number source
      addi $t1, $t1, -1     # decrement loop counter
      bgtz $t1, loop        # repeat if not finished yet.
      la   $a0, fibs        # first argument for print (array)
      add  $a1, $zero, $t5  # second argument for print (size)
      jal  print            # call print routine.
      li   $v0, 10          # system call for exit
      syscall               # we are out of here.

#########  routine to print the numbers on one line.

      .data
space:.asciiz  " "          # space to insert between numbers
head: .asciiz  "The Fibonacci numbers are:\n"
      .text
print:add  $t0, $zero, $a0  # starting address of array
      add  $t1, $zero, $a1  # initialize loop counter to array size
      la   $a0, head        # load address of print heading
      li   $v0, 4           # specify Print String service
      syscall               # print heading
out:  lw   $a0, 0($t0)      # load fibonacci number for syscall
      li   $v0, 1           # specify Print Integer service
      syscall               # print fibonacci number
      la   $a0, space       # load address of spacer for syscall
      li   $v0, 4           # specify Print String service
      syscall               # output string
      addi $t0, $t0, 4      # increment address
      addi $t1, $t1, -1     # decrement loop counter
      bgtz $t1, out         # repeat if not finished
      jr   $ra              # return
```

2.2 Result

After you run the program code, you should get the following result: