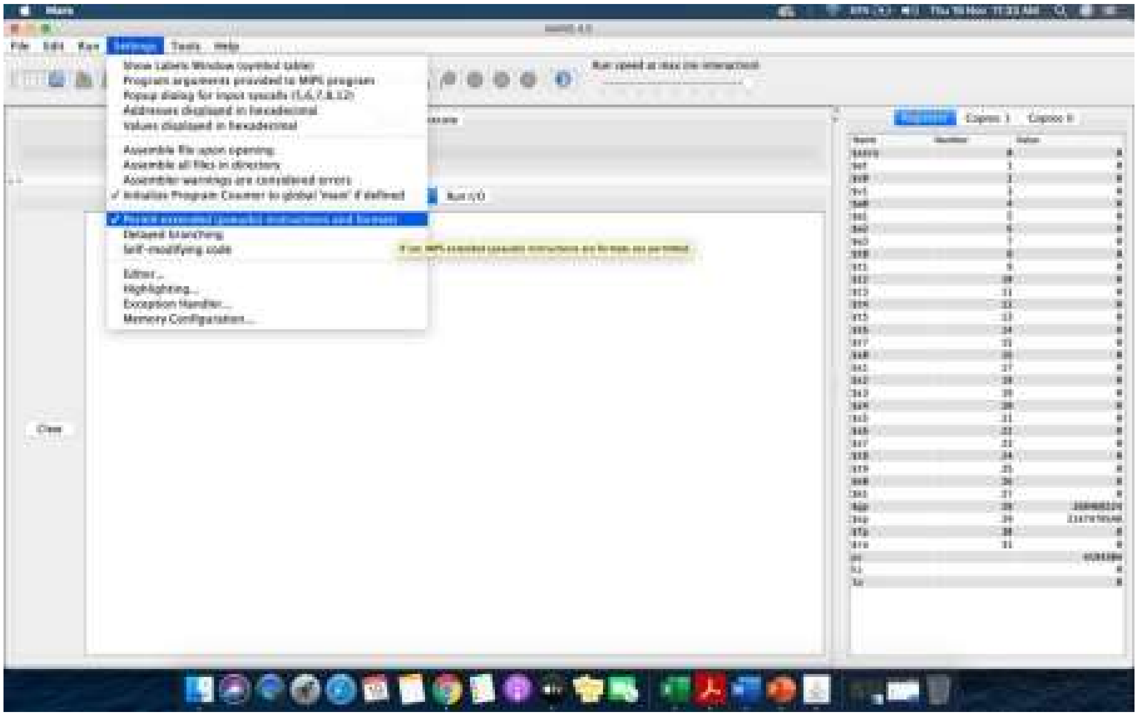
MIPS Programming Manual for COA (15B11CI313)

by

Dr. Hema N

Kindly download the Mar4.5 simulator form
http://courses.missouristate.edu/kenvollmar/mars/

In mars to enable pseudo code, following options to be selected from settings


Mars
MARS 4.5
File
Edit
Run
Settings
Tools
Help
Show Labels Window (symbol table)
Program arguments provided to MIPS program
Popup dialog for input syscalls (5,6,7,8,12)
Addresses displayed in hexadecimal
Values displayed in hexadecimal
Assemble file upon opening
Assemble all files in directory
Assembler warnings are considered errors
✓ Initialize Program Counter to global 'main' if defined
✓ Permit extended (pseudo) instructions and formats
Delayed branching
Self-modifying code
Editor...
Highlighting...
Exception Handler...
Memory Configuration...
If set, MIPS extended (pseudo) instructions are formats are permitted.
Run speed at max (no interaction)
Registers
Coproc 1
Coproc 0
Name
Number
Value
Run I/O
Clear

Example1: ADD

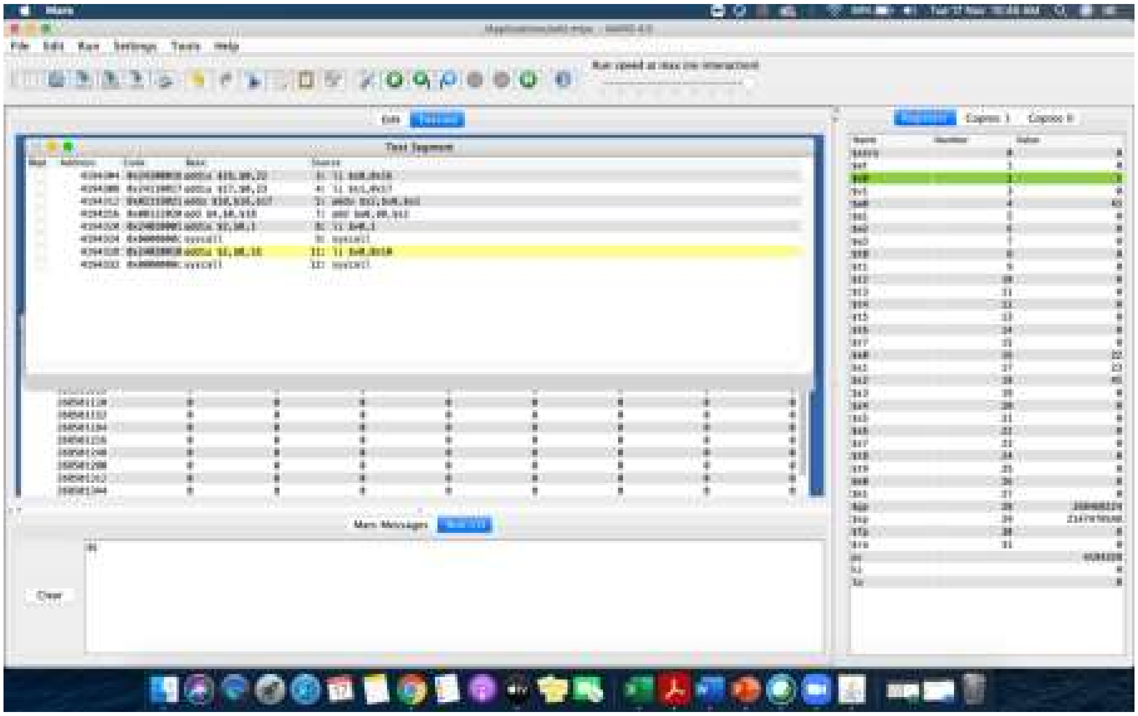
```
 li $t0, 1 #This is to load the immediate value of 1 into the temporary register

$t0  li $t1, 2 #This is to load the immediate value of 2 into the temporary register

 $t1  li $t6, 0 #This is to load the immediate value of 0 into the temporary register $t6
add $t6,$t0,$t1 #this adds the values stored in $t0 and $t1 and assigns them to t6
```

Example 2: SUB

li $t0, 4 #This is to load the immediate value of 4 into the temporary register $t0 li $t1, 2 #This is to load the immediate value of 2 into the temporary register $t1 li $t6, 0 #This is to load the immediate value of 0 into the temporary register $t6 sub $t6,$t0,$t1 #this sub the values stored in $t0 and $t1 and assigns them to t6

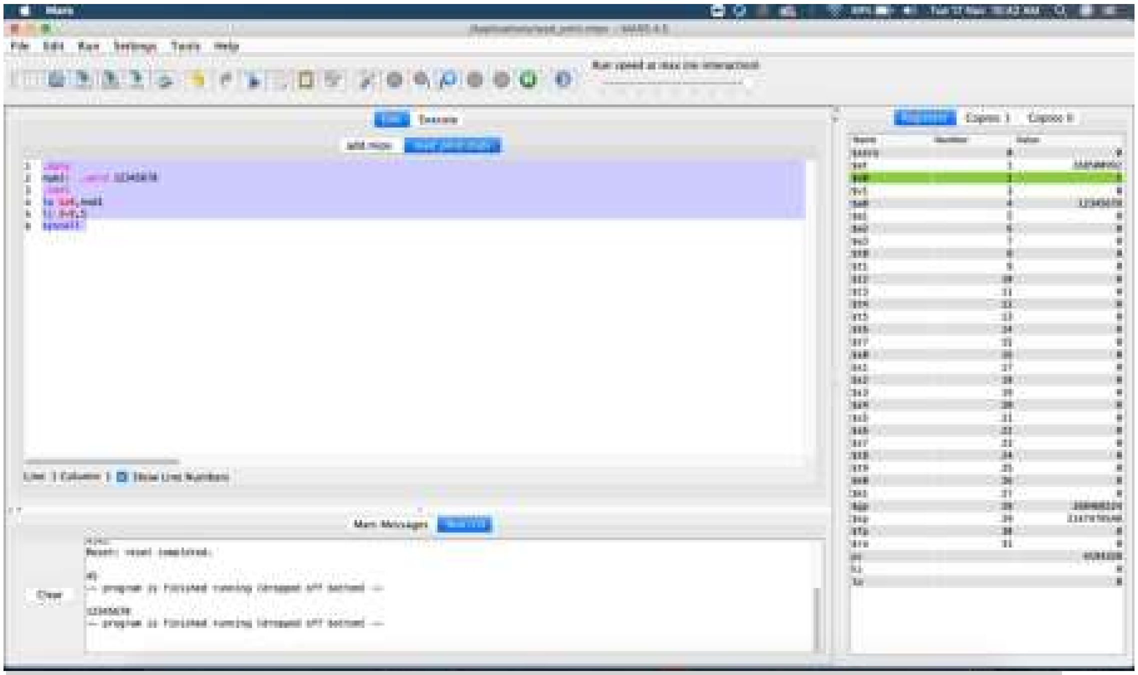
Example 3: Program in MIPS to add two numbers and print on the screen

```
.text
li $s0,0x16
li $s1,0x17
addu $s2,$s0,$s1

add $a0,$0,$s2. #load the sum
li $v0,1
syscall #print on the screen

li $v0,0x10
syscall #Terminate the program
```



Example 4: Program to read word (of size 32-bit) from the memory and print on the screen

```
.data
num1: .word 12345678
.text
lw $a0,num1 # number load into $a0

li $v0,1
syscall #print on the screen
```

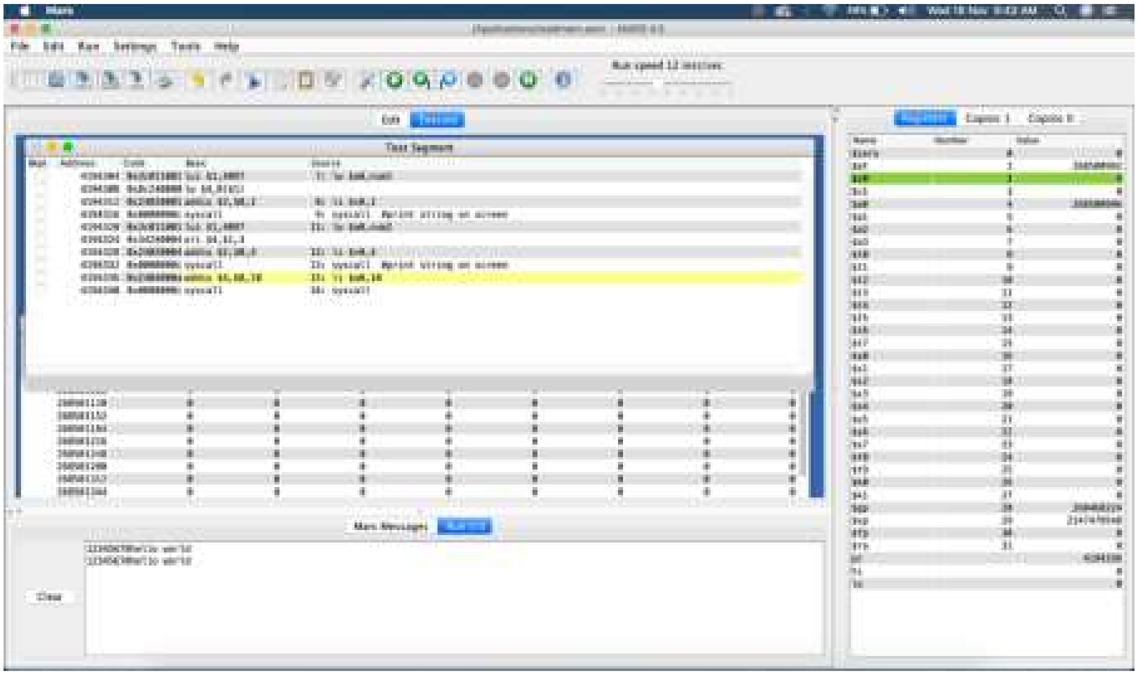


```
.data
num1: .word 12345678
.text
lw $a0,num1
li $v0,1
syscall
```

```
Reset: reset completed.

45
-- program is finished running (dropped off bottom) --

12345678
-- program is finished running (dropped off bottom) --
```

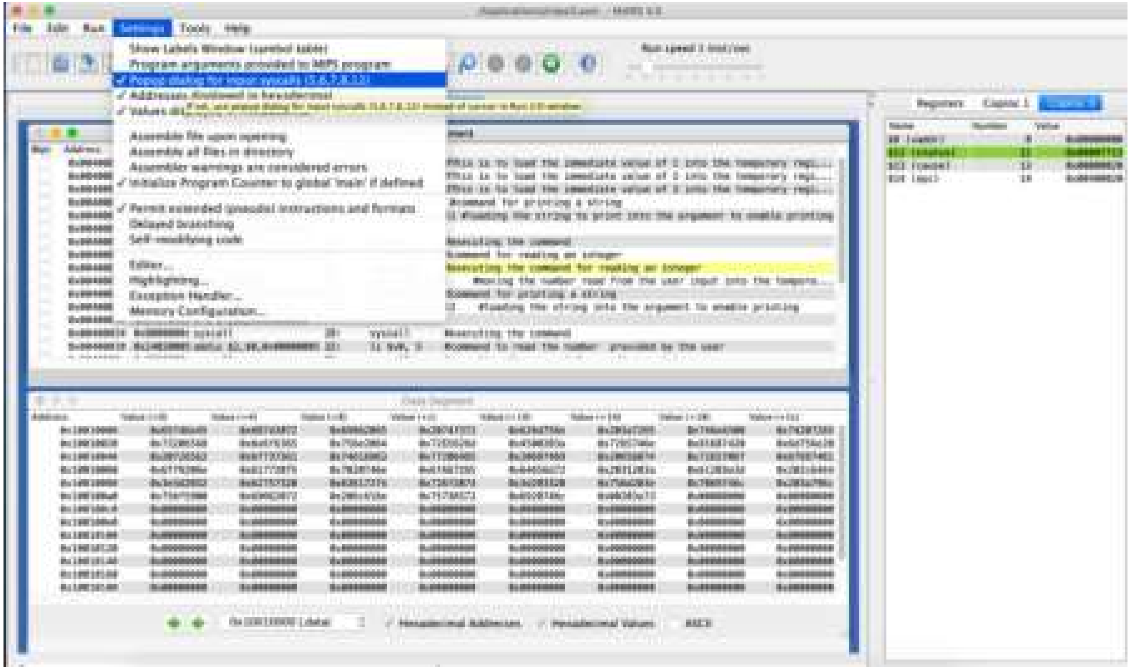
Example 5: Program to read word and string from the memory and print on the screen

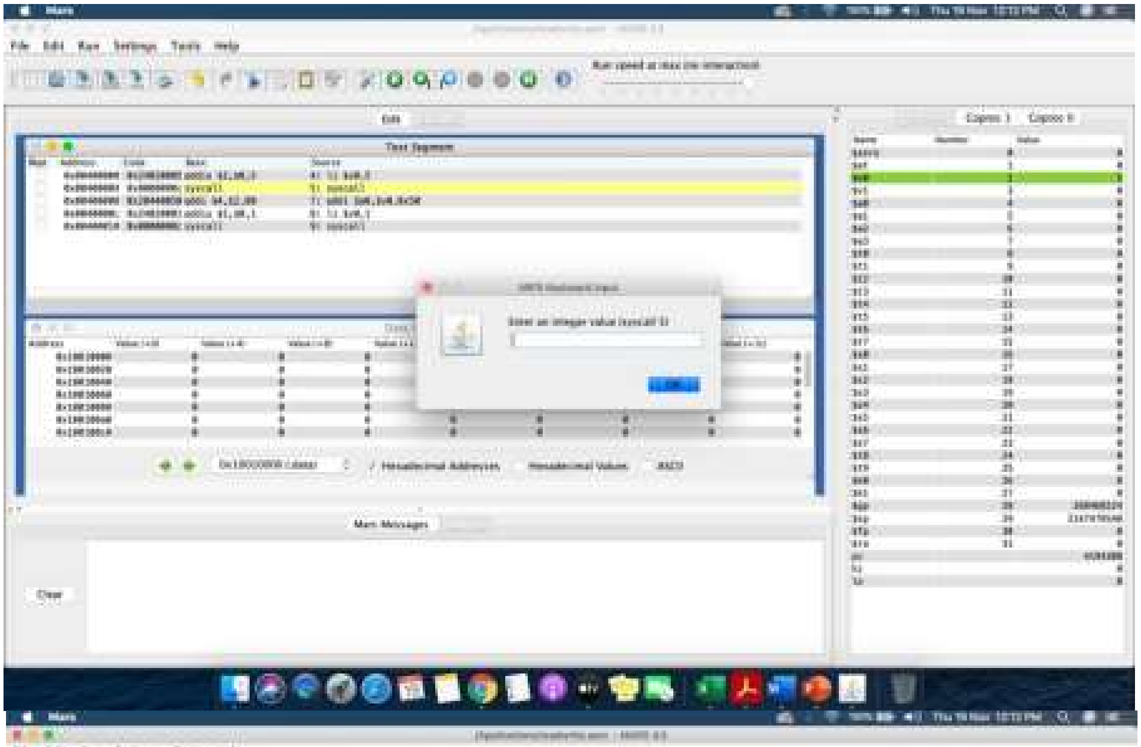
```
.data
num1: .word 12345678
num2: .asciiz "\n hello world \n"

.text

lw $a0,num1
li $v0,1
syscall #print word on screen

la $a0,num2
li $v0,4
syscall #print string on screen


li $a0,10
syscall
```

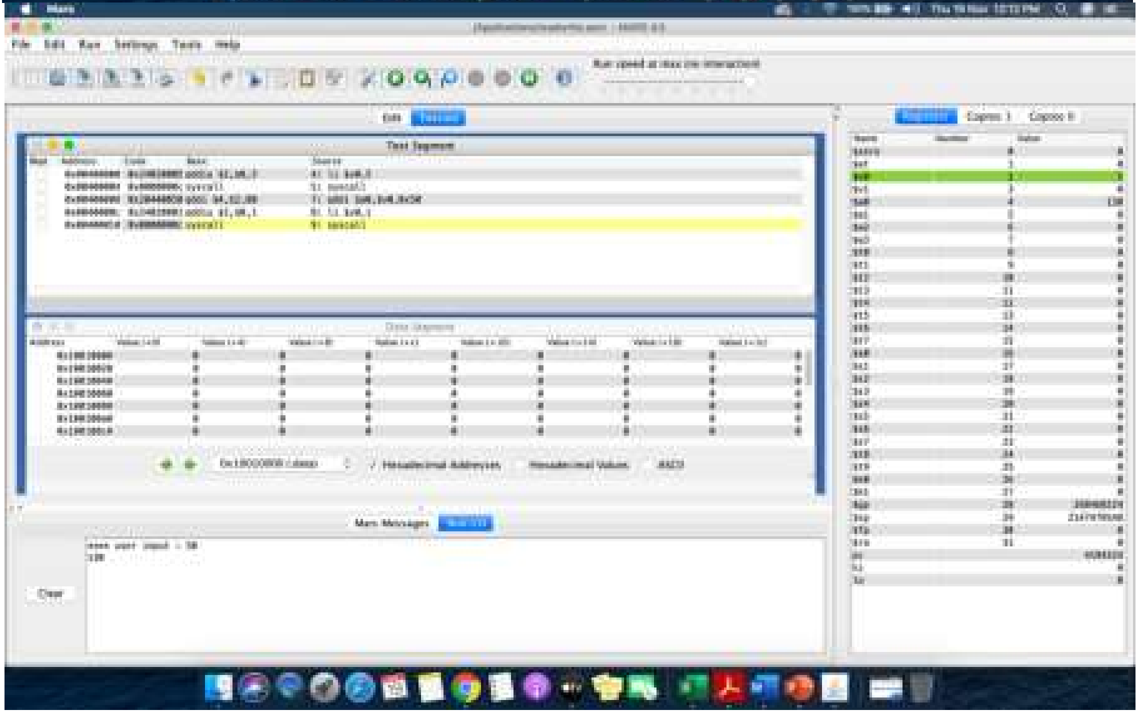


Example 6: To read an integer from the user and add with 50H and print the sum on the screen.

```
.text

li $v0,5
syscall # read number and store at V0

addi $a0,$v0,0x50 # number to printed to be in $a0

li $v0,1
syscall
```


File
Edit
Run
Settings
Tools
Help
Show Labels Window (symbol table)
Program arguments provided to MIPS program
✓ Popup dialog for input syscalls (5,6,7,8,12)
✓ Addresses displayed in hexadecimal
✓ Values displayed in hexadecimal
Assemble file upon opening
Assemble all files in directory
Assembler warnings are considered errors
✓ Initialize Program Counter to global 'main' if defined
✓ Permit extended (pseudo) instructions and formats
Delayed branching
Self-modifying code
Editor...
Highlighting...
Exception Handler...
Memory Configuration...
Data Segment
Hexadecimal Addresses
Hexadecimal Values
ASCII
Registers
Coproc 1
Coproc 0

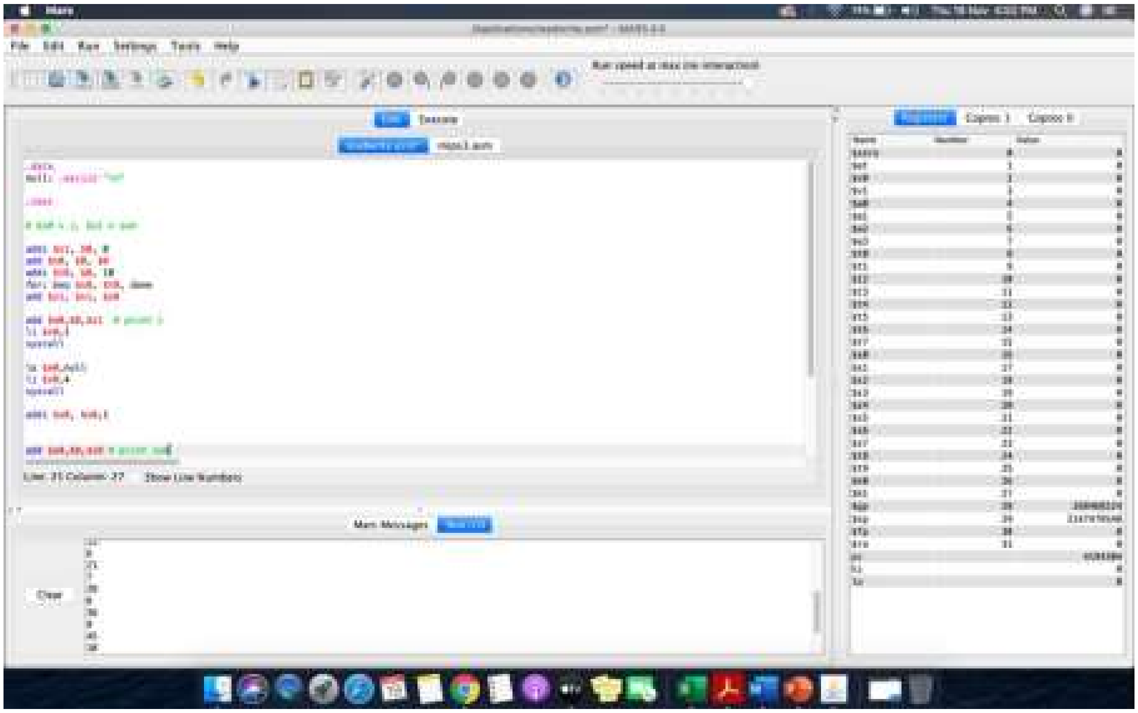




Example 7: Add the number 0 to 9,

```
// addthenumbersfrom 0
to9 intsum= 0;
inti;
for(i = 0; i != 10; i = i+1)
{ sum= sum+ i;
}
```

```
.data
null: .asciiz "\n"

.text

# $s0 = i, $s1 = sum

addi $s1, $0, 0
add $s0, $0, $0
addi $t0, $0, 10
for: beq $s0, $t0, done
add $s1, $s1, $s0

add $a0,$0,$s1 # print i
li $v0,1
syscall



la $a0,null
li $v0,4
syscall

addi $s0, $s0,1


add $a0,$0,$s0 # print sum
li $v0,1
syscall

la $a0,null
li $v0,4
syscall

j for
done:
```



Exercises:

1. **Hello World in MIPS:** Write a MIPS assembly program that prints "Hello, World!" to the console.
2. **Factorial Calculation:** Implement a MIPS program to calculate the factorial of a given number.
3. **Fibonacci Series:** Write a MIPS program to generate the first n terms of the Fibonacci series.
4. **Array Sum:** Create a MIPS program that calculates the sum of elements in an integer array.
5. **Maximum Element in Array:** Implement a MIPS program to find the maximum element in an array.
6. **Array Sorting:** Write a MIPS assembly program to sort an array of integers.
7. **String Length:** Develop a MIPS program to find the length of a null-terminated string.
8. **String Concatenation:** Create a MIPS program to concatenate two strings.
9. **Matrix Multiplication:** Implement a MIPS program to perform matrix multiplication.
10. **Recursive Function:** Write a MIPS program that includes a recursive function, such as computing the factorial recursively.
11. **File Input/Output:** Create a MIPS program that reads data from a file, performs some computation, and writes the result back to another file.
12. **Binary Search Algorithm:** Implement a MIPS program that performs binary search on a sorted array.
13. **Linked List Operations:** Write MIPS code for basic operations on a linked list, such as insertion and deletion.
14. **Procedure Calls:** Create a program with multiple procedures (functions) that are called from the main program.

15. **Bitwise Operations:** Implement MIPS code to perform bitwise operations like AND, OR, XOR, and bit shifting.
16. **Interrupt Handling:** Develop a MIPS program that includes interrupt handling routines.
17. **Memory Manipulation:** Write a program that manipulates memory, demonstrating load and store operations.
18. **Palindrome Check:** Implement a MIPS program to check if a given string is a palindrome.
19. **Bubble Sort:** Write a MIPS program to perform the bubble sort algorithm on an array.
20. **Calculator:** Develop a simple calculator in MIPS that can perform basic arithmetic operations.

Reference:

https://sweetcode.io/building-first-simple-program-mips-assembly-language/
https://courses.cs.washington.edu/courses/cse378/03wi/lectures/mips-asm-examples.html http://www0.cs.ucl.ac.uk/staff/electran/gc03/pdf/07mips_examples.pdf
https://profile.iiita.ac.in/bibhas.ghoshal/COA_2020/Lectures/MIPS_Programming.pdf