

EXERCISE

1. Write a MIPS program to print the string "Hello, MIPS!".

Source Code

```
.data
msg: .asciiz "Hello, MIPS!"
.text
main:
    li $v0, 4
    la $a0, msg
    syscall

    li $v0, 10 # Exit
    syscall
```

Output

```
r data segment [10000000]..[10040000]
000000
010000
010010
```

Console

Hello, MIPS!

```
r Stack
fff798
fff7a0
fff7b0
fff7c0
fff7d0
fff7e0
fff7f0
fff800
fff810
fff820
```

2. Write a MIPS program to print the string "Welcome to MIPS Programming".


Source Code

```
.data
msg: .asciiz "Welcome to MIPS Programming"

.text
.globl main # make 'main' globally visible to linker

main:
    li $v0, 4      # syscall: print_string
    la $a0, msg     # load address of string
    syscall

    li $v0, 10     # syscall: exit
    syscall
```

Output

```
er data segment [10000000]..[10040000]
00000000
00100000
00100100
00100200 Welcome to MIPS Programming

er stack
ffff7980
ffff7a00
```

3. Write a MIPS program to print the character 'A'.

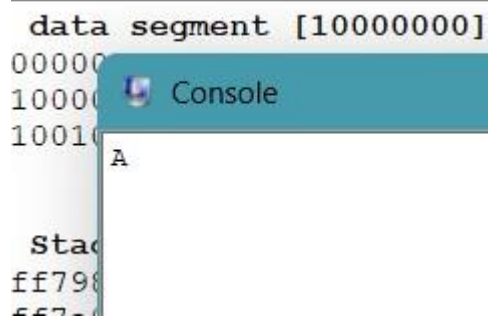
Source Code

```
.data
msg: .asciiz "A"

.text
.globl main
main:
    li $v0, 4
    la $a0, msg
    syscall

    li $v0, 10
    syscall
```

Output



The screenshot shows a MIPS simulator window with a memory dump. The 'data segment' is located at address 10000000. The memory dump shows the following values:

Address	Value
00000000	00000000
10000000	00000000
10010000	00000000

The character 'A' is displayed in the console window.

4. Write a MIPS program to print the character 'B'.

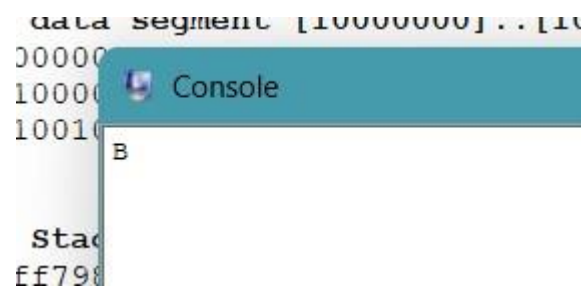
Source Code

```
.data
msg: .asciiz "B"

.text
.globl main
main:
    li $v0, 4
    la $a0, msg
    syscall

    li $v0, 10
    syscall
```

Output



The screenshot shows a console window with a blue title bar labeled "Console". The console output displays the character 'B'. In the background, a memory dump is visible, showing the data segment starting at address 10000000, with addresses 00000, 10000, and 10010 listed. The stack segment is also visible at the bottom, starting at address ff790000.

5. Write a MIPS program to print the string "MIPS" followed by a new line.

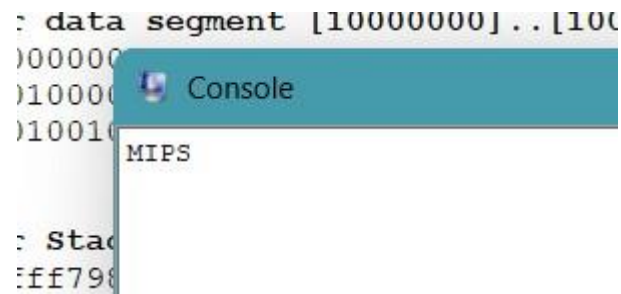
Source Code

```
.data
msg: .asciiz "MIPS\n"

.text
.globl main

main:
    li $v0, 4
    la $a0, msg
    syscall

    li $v0, 10
    syscall
```

Output

The screenshot shows a MIPS simulator interface. On the left, there is a memory dump with addresses 000000, 010000, and 010010. On the right, a console window titled "Console" displays the output "MIPS". Below the console, the stack segment is visible with address 000000 and value 000000.

```
: data segment [10000000]..[10000000]
000000
010000
010010
MIPS
: Stack
000000
000000
```

6. Write a MIPS program to print "First Line" and then print "Second Line" on the next line.

Source Code

```
.data
line1: .asciiz "First Line\n"
line2: .asciiz "Second Line\n"

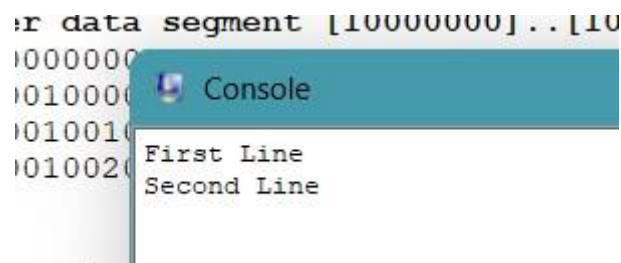
.text
.globl main

main:
    li $v0, 4
    la $a0, line1
    syscall

    li $v0, 4
    la $a0, line2
    syscall

    li $v0, 10
    syscall
```

Output



```
er data segment [10000000]..[10
000000
0010000
0010010
0010020
First Line
Second Line
```

7. Write a MIPS program to print "Hello" on the first line and "World" on the second line.

Source Code

```
.data
hello: .asciiz "Hello\n"
world: .asciiz "World\n"

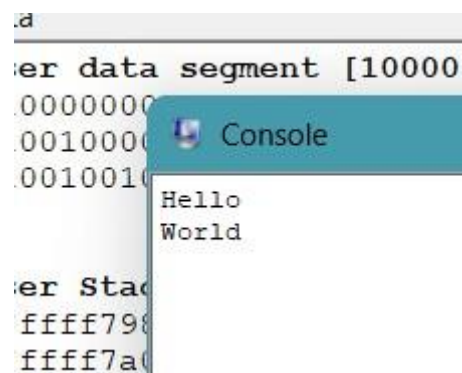
.text
.globl main

main:
    li $v0, 4
    la $a0, hello
    syscall

    li $v0, 4
    la $a0, world
    syscall

    li $v0, 10
    syscall
```

Output



The screenshot shows a MIPS simulator interface. On the left, a memory dump displays addresses and their corresponding values. A console window is open in the center, showing the output of the program. The memory dump includes the data segment starting at 00000000 and the stack segment starting at ffff7980. The console window displays the text "Hello" on the first line and "World" on the second line, matching the program's output.

```
.a
er data segment [10000
.00000000
.00100000
.00100100
er Stac
ffff7980
ffff7a00
```

Console

Hello
World

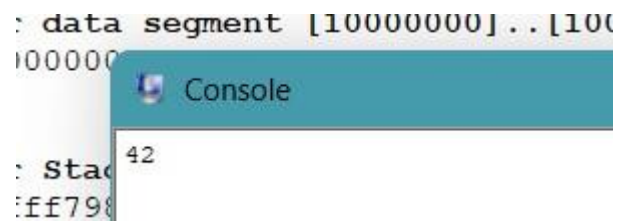
8. Write a MIPS program to print the integer 42.

Source Code

```
.text
.globl main

main:
    li $v0, 1    # print_int
    li $a0, 42
    syscall

    li $v0, 10
    syscall
```

Output

The screenshot shows a console window with a teal header bar labeled "Console". Below the header, the number "42" is displayed, indicating the output of the MIPS program. In the background, a memory dump is visible, showing addresses and hex values, with the value "42" appearing at address "ff798".

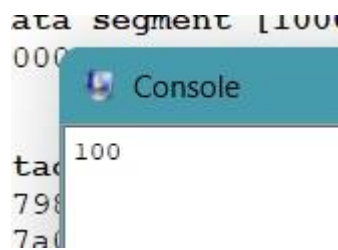
9. Write a MIPS program to print the integer 100.

Source Code

```
.text
.globl main

main:
    li $v0, 1    # print_int
    li $a0, 100
    syscall

    li $v0, 10
    syscall
```

Output

The screenshot shows a console window with a blue header bar labeled "Console". Below the header, the number "100" is displayed, which is the output of the MIPS program. In the background, a portion of a memory dump is visible, showing addresses and hexadecimal values.

10. Write a MIPS program to print the floating-point number 3.14

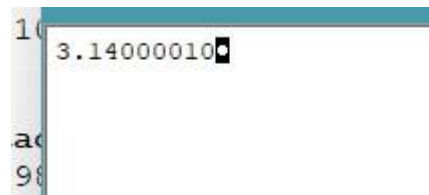
Source Code

```
.data
f: .float 3.14

.text
.globl main

main:
    li $v0, 2      # print_float
    l.s $f12, f
    syscall

    li $v0, 10
    syscall
```

Output

```
10 3.14000010
ac
98
```

11. Write a MIPS program to print the floating-point number 1.23

Source Code

```
.data
f: .float 1.23

.text
.globl main

main:
    li $v0, 2      # print_float
    l.s $f12, f
    syscall

    li $v0, 10
    syscall
```

Output

A screenshot of a terminal window titled "Console". The window shows the output of the MIPS program, which is the floating-point number 1.23000002. The output is displayed on a line that starts with a prompt character, likely a dollar sign, which is partially visible as "10".

```
10 1.23000002
```