

LAB NO. 02: MIPS ASSEMBLY

Objectives:

The objective of this lab is to introduce the fundamentals of MIPS assembly language programming, covering basic syntax, instructions, and the use of registers. In this will learn to implement simple programs using arithmetic operations, data movement, and control flow in MIPS, gaining hands-on experience with low-level programming concepts.

SPIM syscalls:

SPIM provides basic services using the syscall instruction. To request a service, load the system call code into \$v0 and arguments into \$a0 to \$a3 (or \$f12 for floating point values). Results return in \$v0 (or \$f0 for floating points). To display a string, load the string's address into \$a0, set \$v0 to 4, and run syscall. The remaining tasks are defining a string and finding its address.

Task 1:

```
# Name of the programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# task1.asm -- A program that computes the sum of 1 and 2
# result in register $t0
# registers used
# t0 - used to hold the first number
# t1 - used to hold the second number
# t2 - used to hold the result
# v0 - syscall parameter
.text

main:
# SPIM starts execution at main
li $t0,8      # load 8 into $t0
li $t1,5      # load 5 into $t1
add $t2,$t1,$t0    # compute the sum of $t0 and $t1 and put it into $t2
li $v0,10     # syscall code 10 is for exit
syscall       #make the syscall
#end of add.asm
```

Output:

PC		User Text Segment [00400000]..[00440000]	
EPC	= 0	[00400000] 8fa40000 lw \$4, 0(\$29)	; 183: lw \$a0 0(\$sp) # argc
Cause	= 0	[00400004] 27a50004 addiu \$5, \$29, 4	; 184: addiu \$a1 \$sp 4 # argv
BadVAddr	= 0	[00400008] 24a60004 addiu \$6, \$5, 4	; 185: addiu \$a2 \$a1 4 # envp
Status	= 805371664	[0040000c] 00041080 sll \$2, \$4, 2	; 186: sll \$v0 \$a0 2
HI	= 0	[00400010] 00c23021 addu \$6, \$6, \$2	; 187: addu \$a2 \$a2 \$v0
LO	= 0	[00400014] 0c100009 jal 0x00400024 [main]	; 188: jal main
R0 [r0]	= 0	[00400018] 00000000 nop	; 189: nop
R1 [at]	= 0	[0040001c] 3402000a ori \$2, \$0, 10	; 191: li \$v0 10
R2 [v0]	= 10	[00400020] 0000000c syscall	; 192: syscall # syscall 10 (exit)
R3 [v1]	= 0	[00400024] 34080008 ori \$8, \$0, 8	; 13: li \$t0,8 # load 8 into \$t0
R4 [a0]	= 3	[00400028] 34090005 ori \$9, \$0, 5	; 14: li \$t1,5 # load 5 into \$t1
R5 [a1]	= 2147481048	[0040002c] 01285020 add \$10, \$9, \$8	; 15: add \$t2,\$t1,\$t0 # compute the sum of \$t0 and \$t1 and put it into \$t2
R6 [a2]	= 2147481064	[00400030] 3402000a ori \$2, \$0, 10	; 16: li \$v0,10 # syscall code 10 is for exit
R7 [a3]	= 0	[00400034] 0000000c syscall	; 17: syscall #make the syscall
R8 [t0]	= 8	Kernel Text Segment [80000000]..[80010000]	
R9 [t1]	= 5	[80000180] 0001d821 addu \$27, \$0, \$1	; 90: move \$k1 \$at # Save \$at
R10 [t2]	= 13		

Task 2:

Write a program in MIPS assembly language, task2.asm, that computes and prints the sum of two numbers specified at runtime by the user.

```
# Name of the programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# task2.asm-- A program that computes and prints the sum of two numbers.
# Registers used:
#     $t0 -used to hold the first number.
#     $t1 -used to hold the second number.
#     $t2 -used to hold the sum of the $t1 and $t2.
#     $v0 -syscall parameter and return value.
#     $a0 -syscall parameter.
main:
    ## Get first number from user, put into $t0.
    li $v0, 5          # load syscall read_int into $v0.
    syscall            # make the syscall.
    move $t0, $v0      # move the number read into $t0.


    ## Get second number from user, put into $t1.
    li $v0, 5          # load syscall read_int into $v0.
    syscall            # make the syscall.
    move $t1, $v0      # move the number read into $t1.
    add $t2, $t0, $t1   # compute the sum.

    ## Print out $t2.
    move $a0, $t2      # move the number to print into $a0.
    li $v0, 1          # load syscall print_int into $v0.
    syscall            # make the syscall.

    li $v0, 10         # syscall code 10 is for exit.
    syscall            # make the syscall.
# end of task2.asm
```

Output:

R0	[r0]	=	0
R1	[at]	=	0
R2	[v0]	=	10
R3	[v1]	=	0
R4	[a0]	=	5
R5	[a1]	=	2147481048
R6	[a2]	=	2147481064
R7	[a3]	=	0
R8	[t0]	=	3
R9	[t1]	=	2
R10	[t2]	=	5



Data and Text Segment:

The string "Hello World" should not be part of the executable part of the program (which contains all of the instructions to execute), which is called the text segment of the program. Instead, the string should be part of the data used by the program, which is, by convention, stored in the data segment. The MIPS assembler allows the programmer to specify which segment to store each item in a program by the use of several assembler directives. To put something in the data segment, all we need to do is to put a .data before

we define it. Everything between a .data directive and the next .text directive (or the end of the file) is put into the data segment. Note that by default, the assembler starts in the text segment, which is why our earlier programs worked properly even though we didn't explicitly mention which segment to use. In general, however, it is a good idea to include segment directives in your code, and we will do so from this point on.

Task 3:

```
# Programmer Name --Hassan Zaib Jadoon Github: @hzjadoon

.data
a: .asciiz "the answer = "
.text
main:

li $v0, 4    # system call code for printing String
la $a0, a    # address of string to print
syscall      # print the string

li $v0, 1    # system call code for print_int
li $a0, 5    # integer to print
syscall      # print it

li $v0, 10   # system call code for exit
syscall      # exit the program
```

Output:

PC	=	4194368
EPC	=	0
Cause	=	0
BadVAddr	=	0
Status	=	805371664
HI	=	0
LO	=	0
R0 [r0]	=	0
R1 [at]	=	0
R2 [v0]	=	10
R3 [v1]	=	0
R4 [a0]	=	5

Console
the answer = 5

Task 4:

Write a program to take 2 numbers from the user, compute their sum and display the sum result on console

Code:

```
# Name of the programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# task1.asm -- A program that computes the sum of 2 numbers taken from user
# leaving the result in register $t2
# and then print the result on console

main:
    li $v0, 5
    syscall
    move $t0, $v0

    li $v0, 5
    syscall
    move $t1, $v0
    add $t2, $t0, $t1

    move $a0, $t2
    li $v0, 1
    syscall

    li $v0, 10
    syscall
```

Output:

PC	=	4194384	Console
EPC	=	0	
Cause	=	0	4
BadVAddr	=	0	5
Status	=	805371664	9
HI	=	0	
LO	=	0	
R0 [r0]	=	0	
R1 [at]	=	0	
R2 [v0]	=	10	
R3 [v1]	=	0	
R4 [a0]	=	9	
R5 [a1]	=	2147481048	
R6 [a2]	=	2147481064	
R7 [a3]	=	0	
R8 [t0]	=	4	
R9 [t1]	=	5	
R10 [t2]	=	9	

Task 5:

Write a program to take 2 numbers from the user, compute their difference and display the difference result on console

Code:

```
# Name of the programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# task1.asm -- A program that computes the difference of 2 numbers taken from user
# leaving the result in register $t2
# and then print the result on console
main:

li $v0, 5
syscall
move $t0, $v0

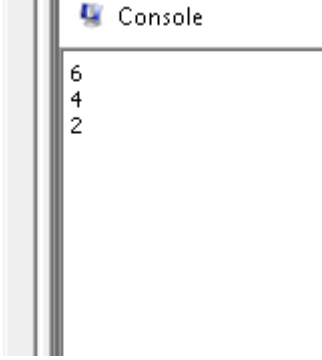
li $v0, 5
syscall
move $t1, $v0
sub $t2, $t0, $t1

move $a0, $t2
li $v0, 1
syscall

li $v0, 10
syscall
```

Output:

R0	[r0]	=	0
R1	[at]	=	0
R2	[v0]	=	10
R3	[v1]	=	0
R4	[a0]	=	2
R5	[a1]	=	2147481048
R6	[a2]	=	2147481064
R7	[a3]	=	0
R8	[t0]	=	6
R9	[t1]	=	4
R10	[t2]	=	2



Console

6
4
2

Task 6:

Write a program to take 2 numbers from the user, compute their product using `mul()` and display the product result on console

Code:

```
# Name of the programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# task1.asm -- A program that computes the product of 2 numbers taken from user
# leaving the result in register $t2
# and then print the result on console

main:
li $v0, 5
syscall
move $t0, $v0

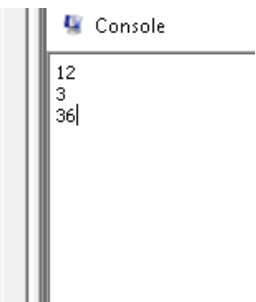
li $v0, 5
syscall
move $t1, $v0
mul $t2, $t0, $t1

move $a0, $t2
li $v0, 1
syscall

li $v0, 10
syscall
```

Output:

R0	[r0]	=	0
R1	[at]	=	0
R2	[v0]	=	10
R3	[v1]	=	0
R4	[a0]	=	36
R5	[a1]	=	2147481048
R6	[a2]	=	2147481064
R7	[a3]	=	0
R8	[t0]	=	12
R9	[t1]	=	3
R10	[t2]	=	36



Task 7:

Write a program to take 2 numbers from the user, compute their division and display the division result on console

Code:

```
# Name of the programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# task1.asm -- A program that computes the division of 2 numbers taken from user
# leaving the result in register $t2
# and then print the result on console
main:

li $v0, 5
syscall
move $t0, $v0

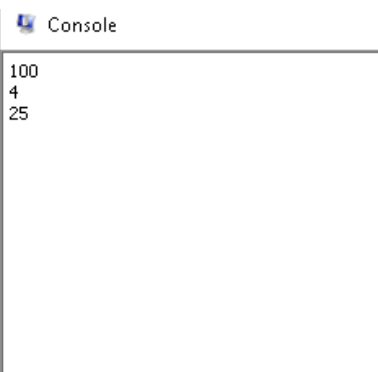
li $v0, 5
syscall
move $t1, $v0
div $t2, $t0, $t1

move $a0, $t2
li $v0, 1
syscall

li $v0, 10
syscall
```

Output:

HI	=	0
L0	=	25
R0	[r0]	= 0
R1	[at]	= 0
R2	[v0]	= 10
R3	[v1]	= 0
R4	[a0]	= 25
R5	[a1]	= 2147481048
R6	[a2]	= 2147481064
R7	[a3]	= 0
R8	[t0]	= 100
R9	[t1]	= 4
R10	[t2]	= 25



Task 8:

Write a program to take 2 numbers from the user, compute their product using `mult()` and display the product result on console

Code:

```
# Name of the programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# task2.asm -- A program that computes the product of 2 numbers taken from user
# loading the result into L0 register and then move it to $t2
# and then print the result on console
main:

li $v0, 5
syscall
move $t0, $v0

li $v0, 5
syscall
move $t1, $v0
mult $t0, $t1
mflo $t2

move $a0, $t2
li $v0, 1
syscall

li $v0, 10
syscall
```

Output:

HI	= 0	
L0	= 12	
R0	[r0]	= 0
R1	[at]	= 0
R2	[v0]	= 10
R3	[v1]	= 0
R4	[a0]	= 12
R5	[a1]	= 2147481048
R6	[a2]	= 2147481064
R7	[a3]	= 0
R8	[t0]	= 3
R9	[t1]	= 4
R10	[t2]	= 12

