MIPS: Branches and Jumps

Aim

The aim of this session is to be able to use the MIPS branch and Jump instructions to implement the equivalent of IF THEN ELSE statements, WHILE and FOR loops and functions/methods/subroutines.

Exercise 1

Enter the program below. When will the program exit?

```
main:
addi $t1, $t1, 1
j main
```

Exercise 2

```
addi $t1,$zero, 20
loop:
addi $t1, $t1, -1
# branch to loop if
# greater than or equal to zero.
bgez $t1, loop
```

The code below loops 20 times. Change the program such that it loops 10 times.

Exercise 3

Modify Exercise 2 such that it prints the numbers as its loops.

The expected output is:

```
109876543210
```

Exercise 4

Modify the code such that it prints a comma between each number.

The expected output is:

```
10,9,8,7,6,5,4,3,2,1,
```

Exercise 5

Modify the code such that its prints the numbers from 1 to 10.

The expected output is:

```
1,2,3,4,5,6,7,8,9,10,
```

Exercise 6

```
.data
strPass: .asciiz "You passed, well done"
strFail: .asciiz "You failed - try harder"

mark: .word 55
.text
lw $s0, mark # load mark from memory into $s3
if:
slti $t0, $s0, 40 # set on less: sets t0 to 1 if S0 is less than 40
beq $t0, $zero, else
then:
la $a0, strFail # set $a0 to the address of the fail string
j endif
else:
la $a0, strPass # set $a0 to the address of the pass string
endif:
# print out the string at address $a0
li $v0, 4
syscall
```

The above code implements a program that prints an appropriate message based on the mark.

Change the program so that the mark is entered by the user and not hardcoded at 55 as it currently is.

Exercise 7

It has been decided that such negative feedback is inappropriate so change the program such that it now prints the following:

0 < Mark < 40: “Nearly there”
40 < Mark < 90: “Well Done”
90 < Mark <100: “Excellent: you can have my job”

Exercise 8

Create a program that request number from the user and prints the factorial of that number.

8 Factorial = 8! = 8*7*6*5*4*3*2*1

Exercise 9

```
.data
menu: .ascii "\n"
.ascii "1, Enter Value of Widgets\n"
.ascii "2, Enter Number of Widgets\n"
.ascii "3, Print Value of Oder\n"
.ascii "12, Quit\n"
.ascii "\n"
.ascii "Please enter menu option "
.asciiz "" MIPS: Branching and Jumps

str.order: .asciiz "The total order cost is "
str.number: .asciiz "How many widgets? "
str.value: .asciiz "Price of each widget? "
.text
addi $s0,$zero, 10 # default Value for Price of a widget
addi $s1,$zero, 5 # default Value for number of widgets required
#Print Menu using service num 4
printMenu: # Label so we can jump/branch back here
addi $v0, $zero, 4
la $a0, menu
syscall
# Read user option using service 5 [num goes into $v0]
addi $v0, $zero, 5
syscall
#move user's choice into $s3
add $s3,$zero, $v0
#process menu
beq $s3,1,opt1
beq $s3,2 opt2
beq $s3,3,opt3
beq $s3,12,quit
#user selected an invalid option so print menu again
j printMenu
opt1:
# do this if user selected 1
addi $v0, $zero, 4
la $a0, str.value
syscall
addi $v0, $zero, 5
syscall
add $s0, $zero, $v0 # set widgetValue(s0) to user enter input(v0)
j printMenu # jump back and print menu again
opt2:
#do this if user selected 2
addi $v0, $zero, 4
la $a0, str.number
syscall
addi $v0, $zero, 5
syscall MIPS: Branching and Jumps
```

```
add $s1, $zero, $v0 # set widgetnum(s1) to user enter input(v0
j printMenu
opt3:
#do this if user selected 3
addi $v0, $zero, 4
la $a0, str.order
syscall
# Calculation order price = number of widgets x value of widget
mul $a0, $s0, $s1
addi $v0, $zero, 1
syscall
j printMenu
quit:
```

The code above [available on the VLE as menu.asm] implements a simple menu system.
Modify the code so that it implements the first 4 the following options:
1, Enter Number 1
2, Enter Number 2
3, Display num1 and num2
4, Display sum of num1 and num2