```
#################################################################################
#ASSEMBLY CALCULATOR
#BERK MUAMMER KUZU
#IDE: MARS
#################################################################################

#################################################################################
.data
string1: .asciiz "Enter the first number: \n"
opChose: .asciiz "Enter the operand: \n"
string2: .asciiz "Enter the second number: \n"
resultMsg: .asciiz "Result: "
errorMsg: .asciiz "Entered wrong operand. Options +, -, *, / \n"
newLine: .asciiz "\n"
carryBit: .asciiz "Remainder: "

op1: .word 1
op2: .word 1
op: .word 2

out: .word 1 #output
remainder: .word 1
#################################################################################

#################################################################################
.text
main:
li $v0, 4
la $a0, string1
syscall #print string 1
```

```
li $v0, 5
syscall #input op1
sw $v0, op1 #save op1 to the memory

opType: #operation function
li $v0, 4
la $a0, opChose
syscall #print opChose

la $a0, op
la $a1, 2
li $v0, 8
syscall #select the operation input
lw $t0, 0($a0)

#####NEW LINE######
li $v0, 4
la $a0, newLine
syscall
####################

##Operand Check##
li $t1, '+'
li $t2, '-'
li $t3, '*'
li $t4, '/'
###################

beq $t0, $t1, secondOp
beq $t0, $t2, secondOp
beq $t0, $t3, secondOp
```

```
beq $t0, $t4, secondOp
j Error

##
secondOp:
li $v0, 4
la $a0, string2
syscall #print string 2
li $v0, 5
syscall
sw $v0, op2

lw $s1, op1
lw $s2, op2
lw $s0, out
lw $s4, remainder
##

####################### + - * / ######################
beq $t0, $t1, addb
beq $t0, $t2, sub_
beq $t0, $t3, mult_
beq $t0, $t4, div_

addb:
add $s0, $s1, $s2
sw $s0, out
j print

sub_:
sub $s0, $s1, $s2
```

```
sw $s0, out
j print

mult_:
mult $s1, $s2
mflo $s0
sw $s0, out
j print

div_:
div $s1, $s2
mflo $s0
mfhi $s4
sw $s4, remainder
li $v0, 4
la $a0, carryBit
syscall #print carryBit
li $v0, 1
lw $a0, remainder
syscall
#####NEW LINE######
li $v0, 4
la $a0, newLine
syscall
####################
sw $s0, out
j print
####################### + - * / ######################

##
print:
```

```
li $v0, 4
la $a0, resultMsg
syscall #print resultMsg

li $v0, 1
lw $a0, out
syscall #print out

#####NEW LINE######
li $v0, 4
la $a0, newLine
syscall
####################

#####NEW LINE######
li $v0, 4
la $a0, newLine
syscall
####################

j main
##

##
Error:
li $v0, 4
la $a0, errorMsg
syscall #print errorMsg
j opType
##
################################################################################
```