**COMPUTER ARCHITECTURE AND ORGANIZATION**

**B S E - 3 A**

**PROJECT REPORT**

**Project Name: UNIT CONVERTOR**

**Submitted to:** **ENGR. Muhammad Rehan Baig**

**Group Members:**

**Syed Muhammad Shoaib (02-131192-030)**

**Raja Humza Nawaz (02-131192-072)**

**Rida Shahab (02-134162-158)**

**Baddiah Javed (02-132182-049)**

**TABLE OF CONTENTS**

[**ABSTRACT:** 2](#_Toc534047216)

[**INTRODUCTION:** 2](#_Toc534047217)

[**ALOGORITHM:** 3](#_Toc534047218)

[**CODE** 5](#_Toc534047218)

[**USER-INTERFACE:** 35](#_Toc534047219)

[**CONCLUSION:** 42](#_Toc534047220)

[**REFERENCES:** 43](#_Toc534047221)

# **ABSTRACT:**

The unit convertor is a powerful tool that provides you with the interface to perform conversions from a given series of units. It is a handy utility for students, teachers, and practitioners in engineering, physics, sciences, and technical subjects. It contains the most important conversion factors for **length, volume, temperature, frequency, plane angle and time units**.

# **INTRODUCTION:**

The Unit Convertor System is command line based system. It is simple and easy to use. It provides results with accuracy in user-friendly mode. This system can be used by business and educational participants. It is to developed with MIPS assembly language, using Mars 4.5.

The system provides user with metrics such as **length, volume, temperature, frequency, plane angle and time conversions**. The user selects the desired metric. Then system provides the user with the menu to select the desired conversion. Once user selects option from the menu then he is asked to enter the value to be converted. In the end result of the conversion is provided and ask whether to continue or exit.

# **ALOGORITHM:**

1) print welcome note

2) display menu of types of conversions (from 1 to 6)

3) take input from the user as an integer and store it

4) if input is out of range generate error message and repeat step 3 else jump to the label according to input selection.

beq $s0,1,length

beq $s0,2,temperature

beq $s0,3,time

beq $s0,4,frequency

beq $s0,5,angle

beq $s0,6,volume

6) display the menu of convertion units range depending on which label it jumped to.

For example if length is selected; show menu of length conversions.

7) take input as an integer and sore it

8) jump to the label according to the input

If length selected:

beq $s1,1,km2m #kilometer to meter conversion

beq $s1,2,km2cm #kilometer to centimeter conversion

beq $s1,3,km2in #kilometer to inch conversion

beq $s1,4,m2km #meter to kilometer conversion

beq $s1,5,m2cm #meter to centimeter conversion

beq $s1,6,m2in #meter to inch conversion

beq $s1,7,cm2km #centimeter to kilometer conversion

beq $s1,8,cm2m #centimeter to meter conversion

beq $s1,9,cm2in #centimeter to inch conversion

beq $s1,10,in2km #inch to kilometer conversion

beq $s1,11,in2m #inch to meter conversion

beq $s1,12,in2cm #inch to centimeter conversion

9)if input is out of the range generate error message and repeat step 7

la $a0,errmsg #generate error message

li $v0,4

syscall

j lengthInput #take input again

10) take input value to be converted as float value

11) make calculations of the conversion

12) display result

13) ask user to continue or exit

14) if user continue jump to step 1

15) main menu appears and the user will be again asked to select

15) else press any other key to terminate the program

# **CODE:**

# **.Data:**

# .data

# welcome: .asciiz "\n\t\tWELCOME TO THE UNIT CONVERTOR\n\n"

# type: .asciiz "Select the input type from the given range:\nFor LENGTH press 1\nFor TEMPERATURE press 2\nFor TIME press 3\nFor FREQUENCY press 4\nFor PLANE ANGLE press 5\nFor VOLUME press 6\n"

# errmsg: .asciiz "Incorrect selection, please try again.\n"

# unitType: .asciiz "Now, Please select the unit conversion;\n"

# len: .asciiz "Kilometer to meter = 1\nKilometer to centimeter = 2\nKilometer to inch = 3\nMeter to kilometer = 4\nMeter to centimeter = 5\nMeter to inch = 6\nCentimeter to Kilometer = 7\nCentimeter to meter = 8\nCentimeter to inch =9\nInch to kilometer = 10\nInch to meter = 11\nInch to centimeter = 12\n"

# tem: .asciiz "Celsius to Fahrenheit = 1\nCelsius to kelvin = 2\nFahrenheit to Celsius = 3\nFahrenheit to kelvin = 4\nKelvin to celsius = 5\nKelvin to fahrenheit = 6\n"

# tim: .asciiz "Second to milisecond = 1\nSecond to minute = 2\nSecond to hour = 3\nMilisecond to second = 4\nMilisecond to minute = 5\nMilisecond to hour = 6\nMinute to second = 7\nMinute to milisecond = 8\nMinute to hour =9\nHour to second = 10\nHour to milisecond = 11\nHour to minute = 12\n"

# frq: .asciiz "Hertz to kilohertz = 1\nHertz to megahertz = 2\nHertz to gigahertz = 3\nKilohertz to hertz = 4\nKilohertz to megahertz = 5\nKilohertz to gigahertz = 6\nMegahertz to hertz = 7\nMegahertz to kilohertz = 8\nMegahertz to gigahertz =9\nGigahertz to hertz = 10\nGigahertz to kilohertz = 11\nGigahertz to megahertz = 12\n"

# angl: .asciiz "Degree to radian = 1\nRadian to degree = 2\n"

# vol: .asciiz "Cubic meter to litre = 1\nCubic meter to mililitre = 2\nLitre to cubic meter = 3\nLitre to mililitre = 4\nMililitre to cubic meter = 5\nMililitre to litre = 6\n""

# val: .asciiz "Enter the value: "

# out: .asciiz "Result of conversion is: "

# exitMsg: .asciiz "\n\nPress 1 to continue and any other key to exit\n"

# pi: .float 3.142

# sixty: .float 60

# oneEighty: .float 180

# hundred: .float 100

# thousand: .float 1000.0

# lac: .float 100000

# million: .float 1000000

# billion: .float 1000000000

# varr1: .float 39370.079

# varr2: .float 39.37

# varr3: .float 2.54

# nineDIVfive: .float 1.8

# fiveDIVnine: .float 0.5555

# num1: .float 32

# num2: .float 273.15

# ########################################## UNITS ##############################################

# kilometer: .asciiz " Km"

# meter: .asciiz " m"

# centimeter: .asciiz " cm"

# inch: .asciiz " in"

# celsius: .asciiz " °C"

# kelvin: .asciiz " K"

# fahrenheit: .asciiz " F"

# second: .asciiz " sec"

# milisecond: .asciiz " ms"

# minute: .asciiz " min"

# hour: .asciiz " hr"

# hertz: .asciiz " Hz"

# kilohertz: .asciiz " KHz"

# megahertz: .asciiz " MHz"

# gigahertz: .asciiz " GHz"

# degree: .asciiz " °"

# raddian: .asciiz " rad"

# cubicmeter: .asciiz " m³"

# litre: .asciiz " l"

# mililitre: .asciiz " ml"

# #s0 is for input type

# #s1 is for selection unit conversion

# #f12 is for result

# **.Text:**

.text

start:

la $a0,welcome

li $v0,4

syscall #welcome note

#giving user selection of input type

la $a0,type

li $v0,4

syscall #input selection note

#reading and saving the input type

input:

li $v0,5 #reading input selection

syscall

move $s0,$v0 #moving selection to s0

#jumping to the conversion type according to the input

beq $s0,1,length

beq $s0,2,temperature

beq $s0,3,time

beq $s0,4,frequency

beq $s0,5,angle

beq $s0,6,volume

#genrating error msg and taking input again if the input is not in range

la $a0,errmsg

li $v0,4

syscall #error msg "wrong selection" note

j input

# **Length Conversion Code:**

########################## CONVERSIONS OF LENGTH #########################

#if length is selected

length:

la $a0,unitType

li $v0,4

syscall #unit type note

la $a0,len

li $v0,4

syscall #units displayed

lengthInput:

li $v0,5

syscall #reading input unit selection

move $s1,$v0 #storing input unit selection in s1

#jumping to desired conversion

beq $s1,1,km2m #kilometer to meter conversion

beq $s1,2,km2cm #kilometer to centimeter conversion

beq $s1,3,km2in #kilometer to inch conversion

beq $s1,4,m2km #meter to kilometer conversion

beq $s1,5,m2cm #meter to centimeter conversion

beq $s1,6,m2in #meter to inch conversion

beq $s1,7,cm2km #centimeter to kilometer conversion

beq $s1,8,cm2m #centimeter to meter conversion

beq $s1,9,cm2in #centimeter to inch conversion

beq $s1,10,in2km #inch to kilometer conversion

beq $s1,11,in2m #inch to meter conversion

beq $s1,12,in2cm #inch to centimeter conversion

la $a0,errmsg #generate error message

li $v0,4

syscall

j lengthInput #take input again

#kilometer to meter

km2m:

jal value #jumping to take value to be converted

l.s $f1,thousand #load 1000 in f1

mul.s $f12,$f0,$f1 #multiply with single precision and store in f12(i.e is ans)

jal result

la $a0,meter

li $v0,4

syscall

j msg

#kilometer to centimeter

km2cm:

jal value

l.s $f1,lac #load 100000 in f1

mul.s $f12,$f0,$f1 #multiply with single precision and store in f12(i.e is ans)

jal result

la $a0,centimeter

li $v0,4

syscall

j msg

#kilometer to inch

km2in:

jal value

l.s $f1,varr1 #load 39370.079 in f1

mul.s $f12,$f0,$f1 #multiply with single precision and store in f12(i.e is ans)

jal result

la $a0,inch

li $v0,4

syscall

j msg

#meter to kilometer

m2km:

jal value #jumping to take value to be converted

l.s $f1,thousand #load 1000 in f1

div.s $f12,$f0,$f1 #Divide with single precision and store in f12(i.e is ans)

jal result

la $a0,kilometer

li $v0,4

syscall

j msg

#meter to centimeter

m2cm:

jal value #jumping to take value to be converted

l.s $f1,hundred #load 100 in f1

mul.s $f12,$f0,$f1 #multiply with single precision and store in f12(i.e is ans)

jal result

la $a0,centimeter

li $v0,4

syscall

j msg

#meter to inch

m2in:

jal value

l.s $f1,varr2 #load 39.37 in f1

mul.s $f12,$f0,$f1 #multiply with single precision and store in f12(i.e is ans)

jal result

la $a0,inch

li $v0,4

syscall

j msg

#centimeter to kilometer

cm2km:

jal value

l.s $f1,lac #load 100000 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e is ans)

jal result

la $a0,kilometer

li $v0,4

syscall

j msg

#cenyimeter to meter

cm2m:

jal value

l.s $f1,hundred #load 100 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e is ans)

jal result

la $a0,meter

li $v0,4

syscall

j msg

#centimeter to inch

cm2in:

jal value #jumping to take value to be converted

l.s $f1,varr3 #load 2.54 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e is ans)

jal result

la $a0,inch

li $v0,4

syscall

j msg

#inch to kilometer

in2km:

jal value

l.s $f1,varr1 #load 39370.079 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e is ans)

jal result

la $a0,kilometer

li $v0,4

syscall

j msg

#inch to meter

in2m:

jal value

l.s $f1,varr2 #load 39.37 in f1

div.s $f12,$f0,$f1 #multiply with single precision and store in f12(i.e is ans)

jal result

la $a0,meter

li $v0,4

syscall

j msg

#inch to centimeter

in2cm:

jal value

l.s $f1,varr3 #load 2.54 in f1

mul.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e is ans)

jal result

la $a0,meter

li $v0,4

syscall

j msg

# **Temperature Conversion Code:**

######################## CONVERSIONS OF TEMPERATURE ##########################

#if temperature is selected

temperature:

la $a0,unitType

li $v0,4

syscall #unit type note

la $a0,tem

li $v0,4

syscall #temperature units displayed

temperatureInput:

li $v0,5

syscall #reading input unit selection

move $s1,$v0 #storing input unit selection in s1

#jumping to desired conversion

beq $s1,1,C2F #celsius to fahrenheit conversion

beq $s1,2,C2K #celsius to kelvin conversion

beq $s1,3,F2C #fahrenheit to celsius conversion

beq $s1,4,F2K #Fahrenheit to kelvin conversion

beq $s1,5,K2C #Kelvin to celsius conversion

beq $s1,6,K2F #Kelvin to fahrenheit conversion

la $a0,errmsg #generate error message

li $v0,4

syscall

j temperatureInput #take the input again

C2F:

jal value

l.s $f1,nineDIVfive #load 1.8 in f1

mul.s $f12,$f0,$f1 #multiply with single precision and store in f12

l.s $f1,num1 #load 32 in f1

add.s $f12,$f12,$f1 #add 32 in muliplied value

jal result

la $a0,fahrenheit

li $v0,4

syscall

j msg

C2K:

jal value

l.s $f1,num2 #load 273.15 in f1

add.s $f12,$f0,$f1 #add 273.15 in value

jal result

la $a0,kelvin

li $v0,4

syscall

j msg

F2C:

jal value

l.s $f1,num1 #load 32 in f1

sub.s $f12,$f0,$f1 #subtract 32 from the value

l.s $f1,fiveDIVnine #load 0.5555 in f1

mul.s $f12,$f12,$f1 #multiply subtraceted value and 0.5555 with single precision and store in f12(i.e is ans)

jal result

la $a0,celsius

li $v0,4

syscall

j msg

F2K:

jal value

l.s $f1,num1 #load 32 in f1

sub.s $f12,$f0,$f1 #subtract 32 from the value

l.s $f1,fiveDIVnine #load 0.5555 in f1

mul.s $f12,$f12,$f1 #multiply subtraceted value and 0.5555 with single precision and store in f12

l.s $f1,num2 #load 273.15 in f1

add.s $f12,$f12,$f1 #add 273.15 in multiplied value with single precision and store in f12(i.e in ans)

jal result

la $a0,kelvin

li $v0,4

syscall

j msg

K2C:

jal value

l.s $f1,num2 #load 273.15 in f1

sub.s $f12,$f0,$f1 #subtract 273.15 from the value and store in f12(i.e ans)

jal result

la $a0,celsius

li $v0,4

syscall

j msg

K2F:

jal value

l.s $f1,num2 #load 273.15 in f1

sub.s $f12,$f0,$f1 #subtract 273.15 from the value and store in f12

l.s $f1,nineDIVfive #load 1.8 in f1

mul.s $f12,$f12,$f1 #multiply subtracted value with 1.8

l.s $f1,num1 #load 32 in f1

add.s $f12,$f12,$f1 #add 32 in multiplied value and store in f12(i.e ans)

jal result

la $a0,fahrenheit

li $v0,4

syscall

j msg

# **Time Conversion Code:**

########################################### CONVERSIONS OF TIME ############################################

#if time is selected

time:

la $a0,unitType

li $v0,4

syscall #unit type note

la $a0,tim

li $v0,4

syscall #units displayed

timeInput:

li $v0,5

syscall #reading input unit selection

move $s1,$v0 #storing input unit selection in s1

#jumping to desired conversion

beq $s1,1,s2ms #second to milisecond conversion

beq $s1,2,s2m #second to minute conversion

beq $s1,3,s2h #second to hour conversion

beq $s1,4,ms2s #milisecond to second conversion

beq $s1,5,ms2m #milisecond to minute conversion

beq $s1,6,ms2h #milisecond to hour conversion

beq $s1,7,m2s #minute to second conversion

beq $s1,8,m2ms ##minute to milisecond conversion

beq $s1,9,m2h ##minute to hour conversion

beq $s1,10,h2s #hour to second conversion

beq $s1,11,h2ms #hour to milisecond conversion

beq $s1,12,h2m #hour to minute conversion

la $a0,errmsg #generate error message

li $v0,4

syscall

j timeInput #take input again

s2ms:

jal value #jumping to take value to be converted

l.s $f1,thousand #load 1000 in f1

mul.s $f12,$f0,$f1 #multiply with single precision and store in f12(i.e ans)

jal result

la $a0,milisecond

li $v0,4

syscall

j msg

s2m:

jal value #jumping to take value to be converted

l.s $f1,sixty #load 60 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,minute

li $v0,4

syscall

j msg

s2h:

jal value #jumping to take value to be converted

l.s $f1,sixty #load 60 in f1

mul.s $f1,$f1,$f1 #load 3600 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,hour

li $v0,4

syscall

j msg

ms2s:

jal value #jumping to take value to be converted

l.s $f1,thousand #load 1000 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,second

li $v0,4

syscall

j msg

ms2m:

jal value #jumping to take value to be converted

l.s $f1,thousand #load 1000 in f1

l.s $f2,sixty #load 60 in f2

mul.s $f1,$f1,$f2 #load 60000 in f1

div.s $f12,$f0,$f1 #value divided by 60000 with single precision and store in f12(i.e ans)

jal result

la $a0,minute

li $v0,4

syscall

j msg

ms2h:

jal value #jumping to take value to be converted

l.s $f1,sixty #load 60 in f1

mul.s $f1,$f1,$f1 #load 3600 in f1

l.s $f2,thousand #load 1000 in f1

mul.s $f1,$f1,$f2 #load 3600000

div.s $f12,$f0,$f1 #value divided by 60000 with single precision and store in f12(i.e ans)

jal result

la $a0,hour

li $v0,4

syscall

j msg

m2s:

jal value #jumping to take value to be converted

l.s $f1,sixty #load 60 in f1

mul.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e is ans)

jal result

la $a0,second

li $v0,4

syscall

j msg

m2ms:

jal value #jumping to take value to be converted

l.s $f1,thousand #load 1000 in f1

l.s $f2,sixty #load 60 in f2

mul.s $f1,$f1,$f2 #load 60000 in f1

mul.s $f12,$f0,$f1 #value divided by 60000 with single precision and store in f12(i.e ans)

jal result

la $a0,milisecond

li $v0,4

syscall

j msg

m2h:

jal value

l.s $f1,sixty #load 60 in f1

div.s $f12,$f0,$f1 #value divided by 60 with single precision and store in f12(i.e ans)

jal result

la $a0,hour

li $v0,4

syscall

j msg

h2s:

jal value #jumping to take value to be converted

l.s $f1,sixty #load 60 in f1

mul.s $f1,$f1,$f1 #load 3600 in f1

mul.s $f12,$f0,$f1 #multiply with single precision and store in f12(i.e ans)

jal result

la $a0,second

li $v0,4

syscall

j msg

h2ms:

jal value #jumping to take value to be converted

l.s $f1,sixty #load 60 in f1

mul.s $f1,$f1,$f1 #load 3600 in f1

l.s $f2,thousand #load 1000 in f1

mul.s $f1,$f1,$f2 #load 3600000

mul.s $f12,$f0,$f1 #value multiplied by 60000 with single precision and store in f12(i.e ans)

jal result

la $a0,milisecond

li $v0,4

syscall

j msg

h2m:

jal value

l.s $f1,sixty #load 60 in f1

mul.s $f12,$f0,$f1 #value multiplied by 60 with single precision and store in f12(i.e ans)

jal result

la $a0,minute

li $v0,4

syscall

j msg

# **Frequency Conversion Code:**

##################################### CONVERSIONS OF FREQURENCY #####################################

#if frequency is selected

frequency:

la $a0,unitType

li $v0,4

syscall #unit type note

la $a0,frq

li $v0,4

syscall #units displayed

frequencyInput:

li $v0,5

syscall #reading input unit selection

move $s1,$v0 #storing input unit selection in s1

#jumping to desired conversion

beq $s1,1,hz2Khz #hertz to kilohertz conversion

beq $s1,2,hz2Mhz #hertz to megahertz conversion

beq $s1,3,hz2Ghz #hertz to gigahertz conversion

beq $s1,4,Khz2hz #kilohertz to hertz conversion

beq $s1,5,Khz2Mhz #kilohertz to megahertz conversion

beq $s1,6,Khz2Ghz #kilohertz to gigahertz conversion

beq $s1,7,Mhz2hz #megahertz to hertz conversion

beq $s1,8,Mhz2Khz ##megahertz to kilohertz conversion

beq $s1,9,Mhz2Ghz ##megahertz to gigahertz conversion

beq $s1,10,Ghz2hz #gigahertz to hertz conversion

beq $s1,11,Ghz2Khz #gigahertz to kilohertz conversion

beq $s1,12,Ghz2Mhz #gigahertz to megahertz conversion

la $a0,errmsg #generate error message

li $v0,4

syscall

j frequencyInput #take input again

hz2Khz:

jal value #jumping to take value to be converted

l.s $f1,thousand #load 1000 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,kilohertz

li $v0,4

syscall

j msg

hz2Mhz:

jal value #jumping to take value to be converted

l.s $f1,million #load 1000000 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,megahertz

li $v0,4

syscall

j msg

hz2Ghz:

jal value #jumping to take value to be converted

l.s $f1,billion #load 1000000000 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,gigahertz

li $v0,4

syscall

j msg

Khz2hz:

jal value #jumping to take value to be converted

l.s $f1,thousand #load 1000 in f1

mul.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,hertz

li $v0,4

syscall

j msg

Khz2Mhz:

jal value #jumping to take value to be converted

l.s $f1,thousand #load 1000 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,megahertz

li $v0,4

syscall

j msg

Khz2Ghz:

jal value #jumping to take value to be converted

l.s $f1,million #load 1000000 in f1

div.s $f12,$f0,$f1 #value divided by 1000000 with single precision and store in f12(i.e ans)

jal result

la $a0,gigahertz

li $v0,4

syscall

j msg

Mhz2hz:

jal value #jumping to take value to be converted

l.s $f1,million #load 1000000 in f1

mul.s $f12,$f0,$f1 #value multiplied by 1000000 with single precision and store in f12(i.e ans)

jal result

la $a0,hertz

li $v0,4

syscall

j msg

Mhz2Khz:

jal value #jumping to take value to be converted

l.s $f1,thousand #load 1000 in f1

mul.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,kilohertz

li $v0,4

syscall

j msg

Mhz2Ghz:

jal value #jumping to take value to be converted

l.s $f1,thousand #load 1000 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,gigahertz

li $v0,4

syscall

j msg

Ghz2hz:

jal value #jumping to take value to be converted

l.s $f1,billion #load 1000000000 in f1

mul.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,hertz

li $v0,4

syscall

j msg

Ghz2Khz:

jal value #jumping to take value to be converted

l.s $f1,million #load 1000000 in f1

mul.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,kilohertz

li $v0,4

syscall

j msg

Ghz2Mhz:

jal value #jumping to take value to be converted

l.s $f1,thousand #load 1000 in f1

mul.s $f12,$f0,$f1 #divide with single precision and store in f12(i.e ans)

jal result

la $a0,megahertz

li $v0,4

syscall

j msg

# **Plane Angle Conversion Code:**

######################## CONVERSIONS OF PLANE ANGLE ##########################

#if plane angle is selected

angle:

la $a0,unitType

li $v0,4

syscall #unit type note

la $a0,angl

li $v0,4

syscall #volume units displayed

angleInput:

li $v0,5

syscall #reading input unit selection

move $s1,$v0 #storing input unit selection in s1

#jumping to desired conversion

beq $s1,1,d2r #degree to radian conversion

beq $s1,2,r2d #radianto degree conversion

la $a0,errmsg

li $v0,4

syscall

j angleInput

#degree to radian

d2r:

jal value

l.s $f1,pi #load value of pi in f1

l.s $f2,oneEighty #load 180 in f2

div.s $f1,$f1,$f2

mul.s $f12,$f0,$f1 #multiply with single precision and store in f12

jal result

la $a0,raddian

li $v0,4

syscall

j msg

r2d:

jal value

l.s $f1,oneEighty #load 180 in f1

l.s $f2,pi #load value of pi in f1

div.s $f1,$f1,$f2

mul.s $f12,$f0,$f1 #multiply with single precision and store in f12

jal result

la $a0,degree

li $v0,4

syscall

j msg

# **Volume Conversion Code:**

######################## CONVERSIONS OF VOLUME ##########################

#if volume is selected

volume:

la $a0,unitType

li $v0,4

syscall #unit type note

la $a0,vol

li $v0,4

syscall #volume units displayed

volumeInput:

li $v0,5

syscall #reading input unit selection

move $s1,$v0 #storing input unit selection in s1

#jumping to desired conversion

beq $s1,1,c2l #cubic metre to litre conversion

beq $s1,2,c2ml #cubic metre to mililitre conversion

beq $s1,3,l2c #litre to cubic metre conversion

beq $s1,4,l2ml #litre to mililitre conversion

beq $s1,5,ml2c #mililitre to cubic metre conversion

beq $s1,6,ml2l #mililitre to litre conversion

la $a0,errmsg

li $v0,4

syscall

j volumeInput

#cubic metre to litre

c2l:

jal value

l.s $f1,thousand #load 1000 in f1

mul.s $f12,$f0,$f1 #multiply with single precision and store in f12

jal result

la $a0,litre

li $v0,4

syscall

j msg

#cubic metre to mililitre

c2ml:

jal value

l.s $f1,million #load value of million in f1

mul.s $f12,$f0,$f1 #multiply with single precision and store in f12(i.e ans)

jal result

la $a0,mililitre

li $v0,4

syscall

j msg

#litre to cubic metre

l2c:

jal value

l.s $f1,thousand #load 1000 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12

jal result

la $a0,cubicmeter

li $v0,4

syscall

j msg

#litre to mililitre

l2ml:

jal value

l.s $f1,thousand #load 1000 in f1

mul.s $f12,$f0,$f1 #divide with single precision and store in f12

jal result

la $a0,mililitre

li $v0,4

syscall

j msg

#mililitre to cubic metre

ml2c:

jal value

l.s $f1,million #load value of million in f1

div.s $f12,$f0,$f1 #multiply with single precision and store in f12(i.e ans)

jal result

la $a0,cubicmeter

li $v0,4

syscall

j msg

#mililitre to litre

ml2l:

jal value

l.s $f1,thousand #load 1000 in f1

div.s $f12,$f0,$f1 #divide with single precision and store in f12

jal result

la $a0,litre

li $v0,4

syscall

j msg

# **Messages:**

######################################################################################################################

value:

la $a0,val

li $v0,4

syscall #msg to enter value

li $v0,6 #read float

syscall

jr $ra #return the the function

result:

la $a0,out

li $v0,4

syscall #msg for result

li $v0,2

syscall #print float

jr $ra

msg:

la $a0,exitMsg

li $v0,4

syscall #msg for exit/continue

li $v0,5

syscall #reading selection(exit/continue)

beq $v0,1,start #continue

j exit #terminate

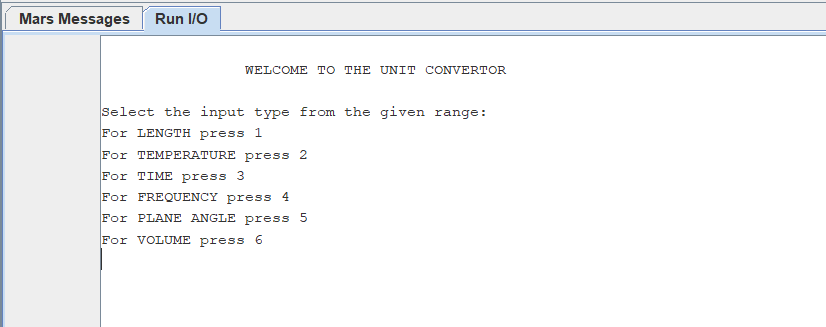
exit:

li $v0,10

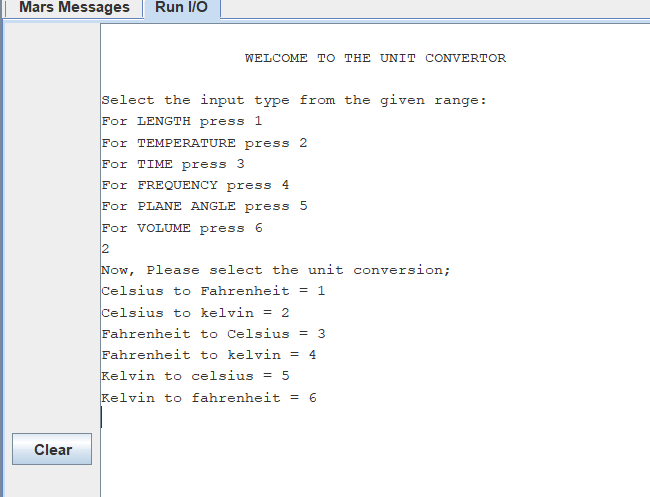
syscall

# **USER-INTERFACE:**

**Starting window**

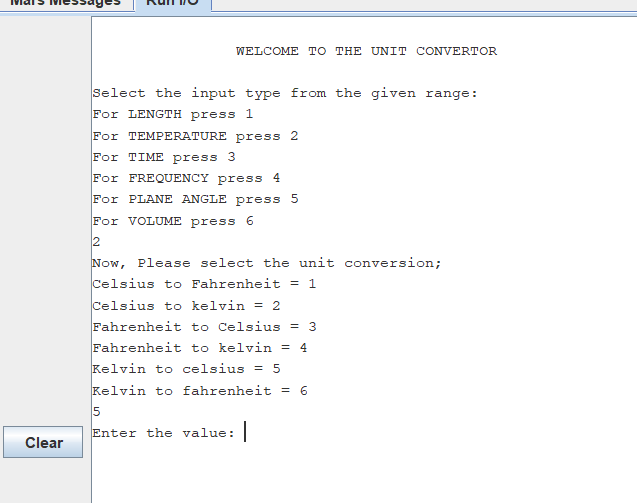


**Input type selected**

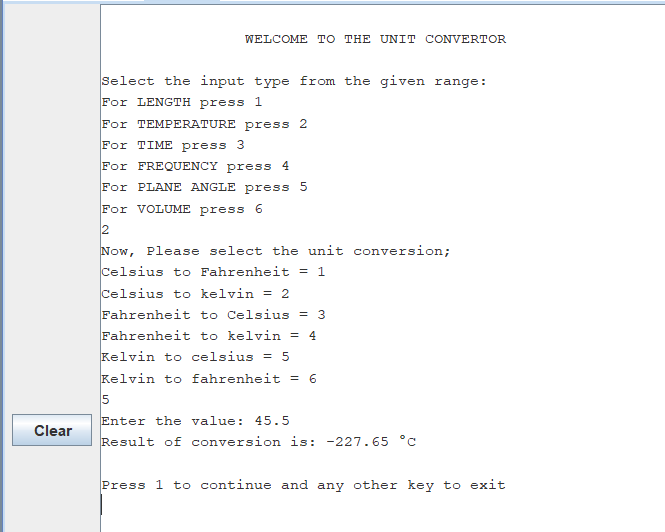


**Unit conversion selected**

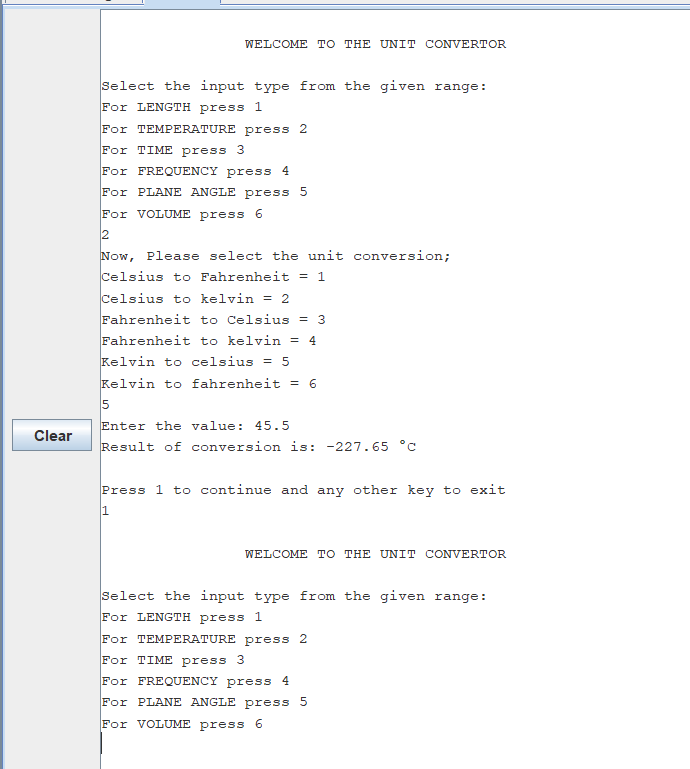
**TEMPERATURE CONVERSION:**



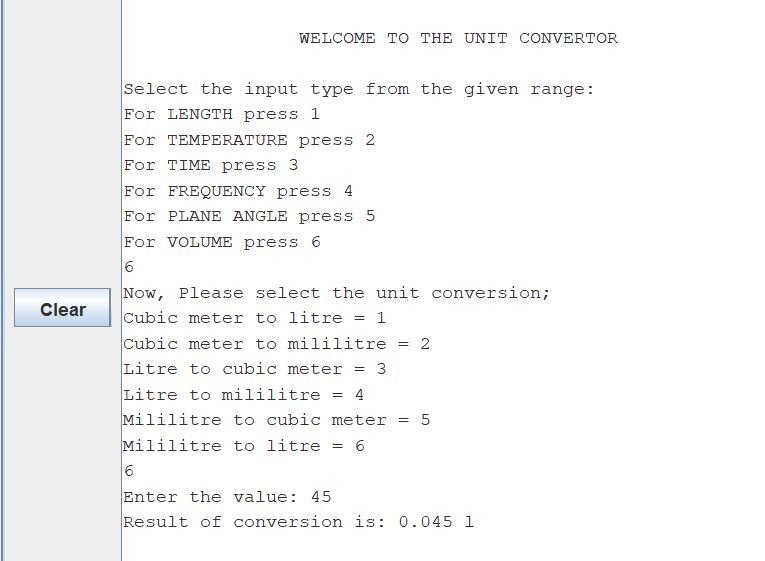
**Value to be converted is entered and result is shown**



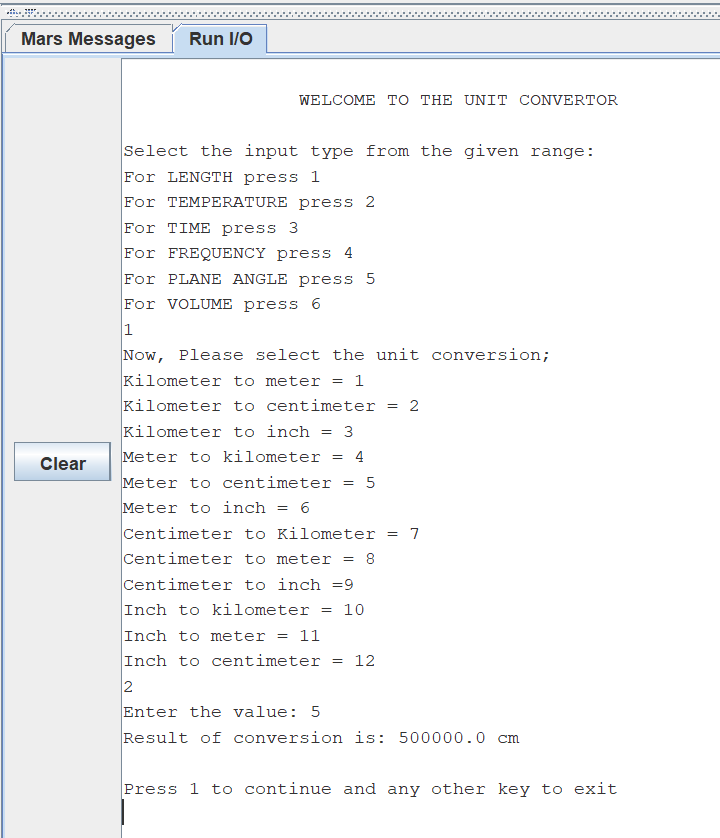
**Conversion continued**



**VOLUME CONVERSION:**

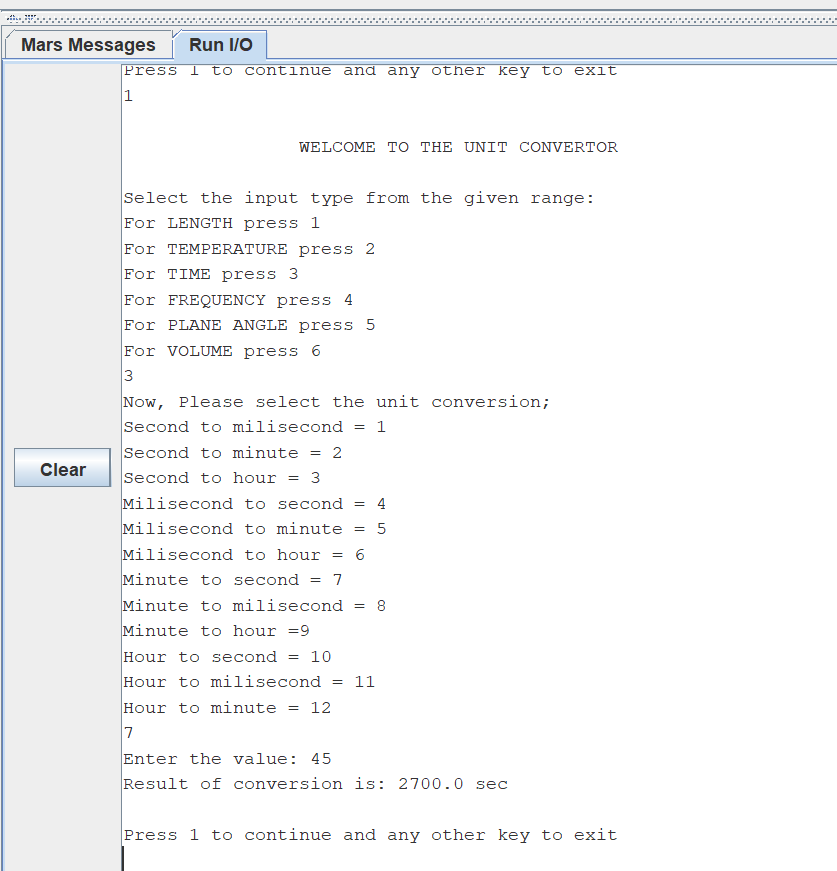


**LENGTH CONVERSION:**

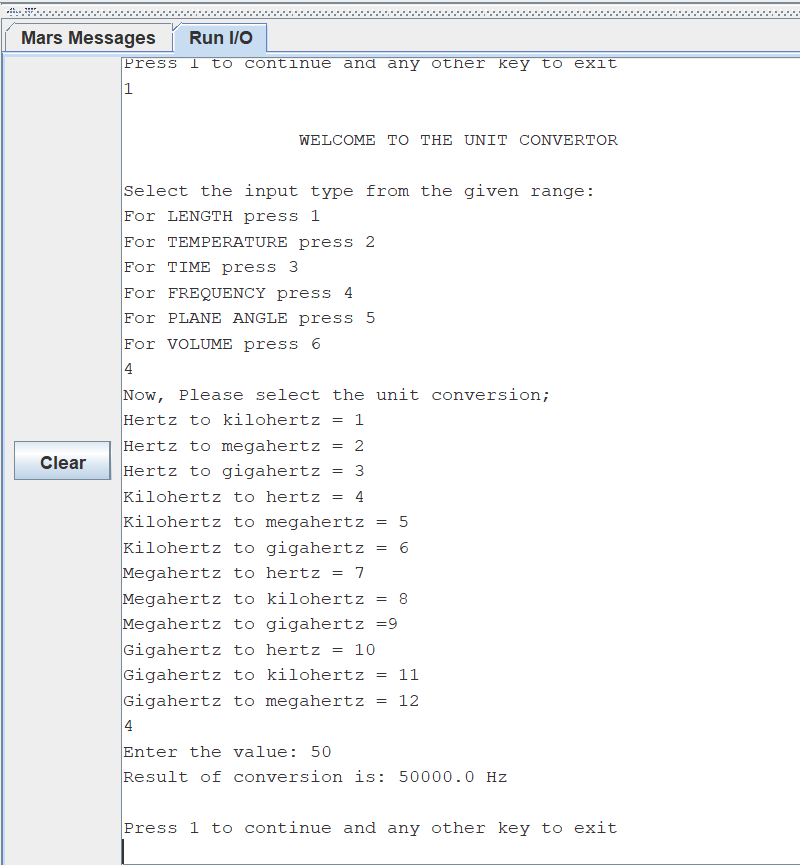


**Conversion continued**

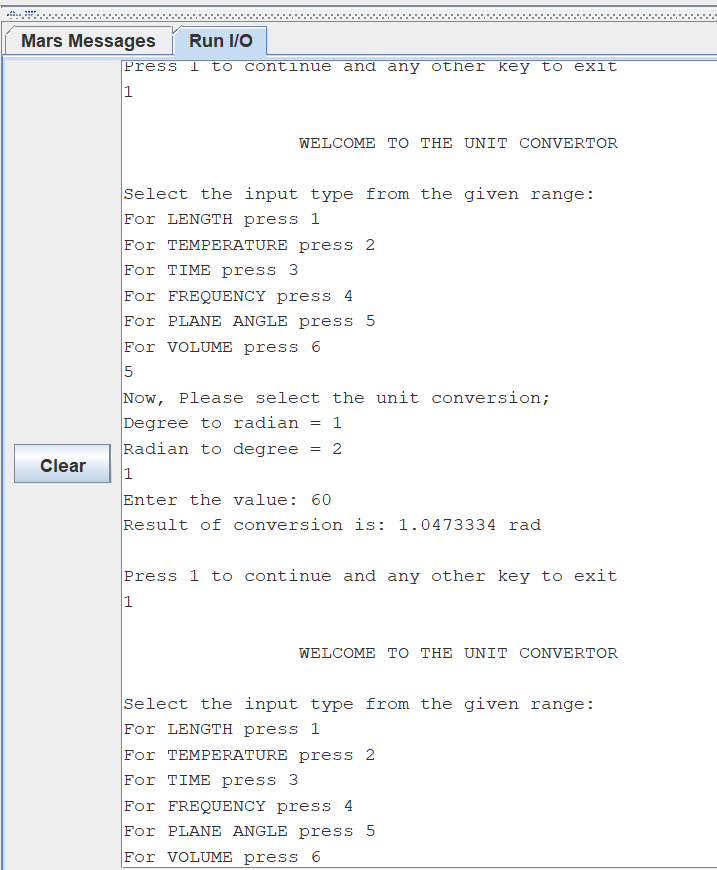
**TIME CONVERSION:**

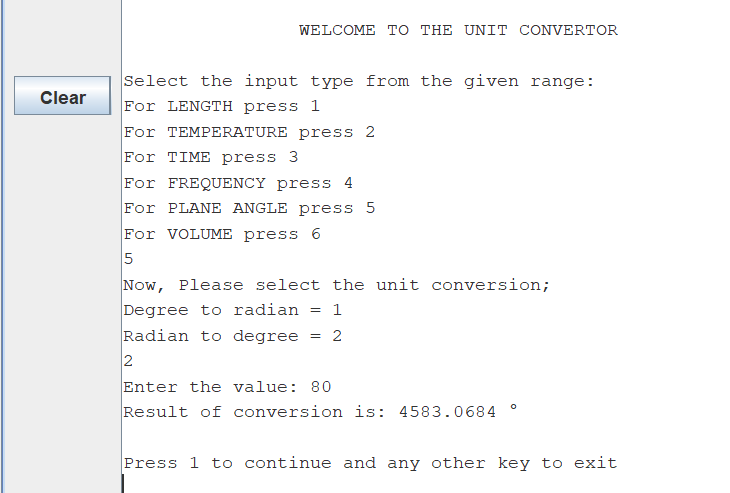


**FREQUENCY CONVERSION:**

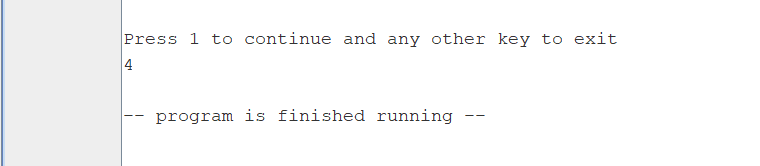


**PLANE ANGLE CONVERSION:**





**EXIT FROM PROGRAM:**



# **CONCLUSION:**

Unit convertor allows you to get instant and accurate results of conversions on hands. It provides various unit conversion on one platform. It is used instead of manually calculating long and difficult conversion. It is a friendly system that can be used by any user.

# **REFERENCES:**

* <https://www.unitconverters.net/volume-converter.html>
* <https://www.onlineconversion.com/frequency.htm>
* <https://www.calculatorsoup.com/calculators/conversions/angle.php>
* <https://www.calculator.net/conversion-calculator.html>
* <https://www.calculatorsoup.com/calculators/conversions/temperature.php>
* <https://www.calculatorsoup.com/calculators/conversions/time.php>