

# Sean O'Mahoney (16042079)

Computer Systems Fundamentals - 6G4Z1102

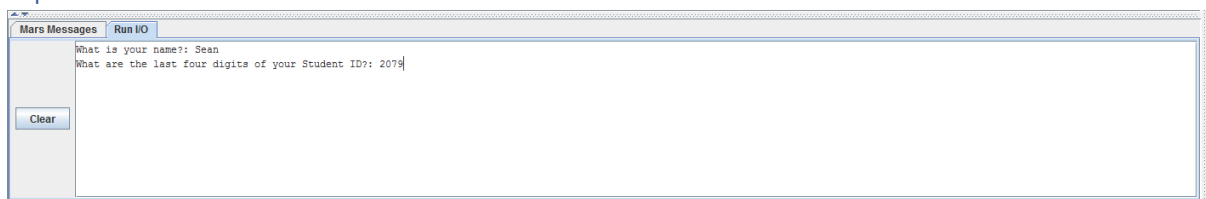
## Task A

At the beginning of the program, there are some string variables, I then start by loading the string to ask the user for their input (showing the string). Then I move on to get an input from them, for their name, I load the argument registers to store.

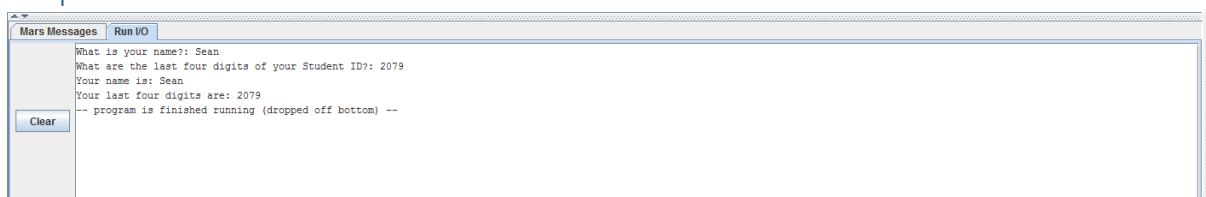
After this, I move onto the retrieval of their ID, which uses the same display of a string, although then moves that into a t register.

Once this is done, I display them to the user, exactly how the strings were displayed previously, although with a syscall of 1 for the ID.

## Input



## Output



## Code

```
.data
name: .space 15

enterName: .asciiz "What is your name?: "
enterNumber: .asciiz "What are the last four digits of your Student ID?: "

resultName: .asciiz "Your name is: "
resultNumber: .asciiz "Your last four digits are: "

.text

#Name
li $v0, 4 #Load a string
la $a0, enterName #Puts string declared into index a0
syscall #Prints

li $v0, 8 #Load String
la $a0, name
li $a1, 15 #Max size of 15
```

```

syscall

#ID
li $v0, 4 #Load a string
la $a0, enterNumber #Puts string declared into index a0
syscall #Prints

li $v0, 5 #Get integer type
syscall #Gets

move $t0, $v0 #Move input to stored index

#Display

#Name
li $v0, 4 #Loads a string
la $a0, resultName
syscall

li $v0, 4 #Load string type
la $a0, name #Moves name into display index
syscall #Shows

#ID
li $v0, 4 #Loads a string type
la $a0, resultNumber
syscall

li $v0, 1 #Load number
move $a0, $t0 #Moves input into display index
syscall #Shows

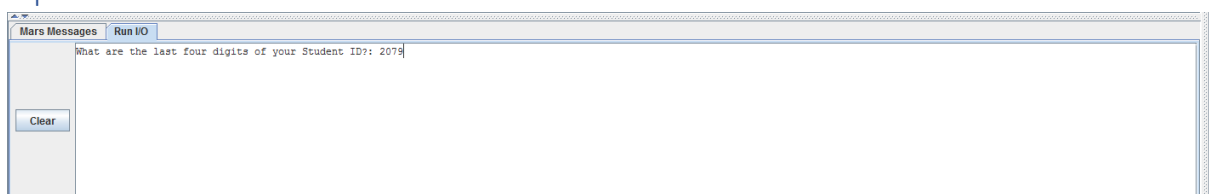
```

## Task B

The program loads a string, which is loaded from .data, to ask for an input. Then a value of 5 is used in \$v0 to get an integer, which is then moved to be stored in \$t0 (which won't be overwritten).

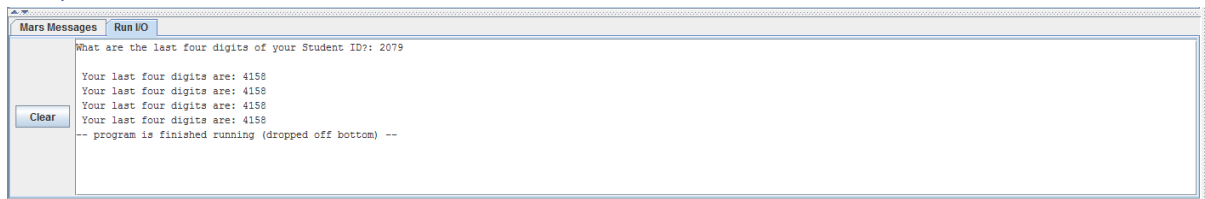
This number is then multiplied by 2, then stored in \$t1 from the LO register, before moving onto the loop method. This method breaks into the end when the \$t3 register is 3 (from 0), increments it if not, displays a string to declare the number, then the number, before jumping back to the start of the loop method.

## Input



The screenshot shows a window titled 'Mars Messages' with a 'Run I/O' button. The main area contains the text 'What are the last four digits of your Student ID?: 2079'. There is a 'Clear' button on the left side of the window.

## Output



## Code

```
.data
enterNumber: .asciiz "What are the last four digits of your Student ID?: "
resultNumber: .asciiz "\n Your last four digits are: "

.text
#Get ID
li $v0, 4 #Load a string
la $a0, enterNumber #Puts string declared into index a0
syscall #Prints

li $v0, 5 #Get integer type
syscall #Gets

move $t0, $v0 #Move input to stored index

#Multiply ID
li $t2, 2 #Sets 2 in t2 reg'
mult $t0, $t2 #Multiply by 2
mflo $t1 #Move result to t1 reg'

#Loops 4 times, t3 reg 0 to 3lki
loop:
    bgt $t3, 3, exit #if t4 reg is 3, move to exit method
    addi $t3, $t3, 1 #adds 1 to t4 reg each loop

#Display ID
li $v0, 4 #Loads a string type
la $a0, resultNumber
syscall

li $v0, 1 #Load number
move $a0, $t1 #Moves input into display index
syscall #Shows

j loop #jump back to loop method

#End of Program
exit:
```

## Task C

Firstly, the program prepares then loads a string, then prints it to the screen, which is then followed by getting an integer variable, then moving it to a stored register. Once that is done, the current time will be loaded, then partly moved into a stored register (since only \$a0 is needed), before moving onto the loop method.

If the \$t3 register is equal to the users input (stored in \$t1), then it will branch to the display method, which is after the nested loops, that are explained next. Once that has been done it will add to the register it has tested (\$t3), then increments it, before resetting the inner loops register to 0 (\$t4).

The second loop is called similarly, branching to the first loop if it's register has reached 10, although if not, increments and resets the inner, before moving to it.

Then there is the third nested loop, which does the same as above, but testing if reached 500. This doesn't have an inner loop to reset, but does a simple multiplication, which is then moved to a stored register.

Meaning, the inner loop will 500 times, the upper loop 10 times (totalling 5000 times), then the beginning loop of the nested loops, will loop it again based on the user input (if the user inputs 910, then it will loop 4,550,000 times entirely).

Once the display method is called, it will get the current time again (using the same code, but storing in \$t7, instead of \$t6), display the result of the multiplication, show the string for "Time Elapsed", then calculates. The calculation is done by preparing to display an integer, then that integer will be the subtraction of the latest time from the start time (after user input at the beginning of the program).

## Code

```
.data
inputText: .ascii "Please enter a three digit number (900-999): "
timeElapsed: .ascii "\nTime Elapsed (in Miliseconds): "

.text
#Get Number Text
li $v0, 4 #Load a string
la $a0, inputText #Puts string declared into index a0
syscall #Prints

#Gets Number
li $v0, 5 #Get integer type
syscall #Gets
move $t1, $v0 #Move input to stored index

#Get start time
li $v0, 30 #gets a 64-bit timestamp
syscall
move $t6, $a0 #Move time to $t7

loop:
    bgt $t3, $t1, display #if t3 reg is the user input, move to display method
```

```
addi $t3, $t3, 1 #adds 1 to t3 reg each loop
li $t4, 0
```

```
loop2:
    bgt $t4, 10, loop
    addi $t4, $t4, 1
    li $t5, 0
```

```
loop3:
    bgt $t5, 500, loop2
    addi $t5, $t5, 1
```

```
#Multiply ID
li $t2, 7
div $t1, $t2
mflo $t0 #Move result to t0 reg'
```

```
j loop3
```

```
display:
```

```
#Get end time
li $v0, 30 #gets a 64-bit timestamp
syscall
move $t7, $a0 #Move time to $t7
```

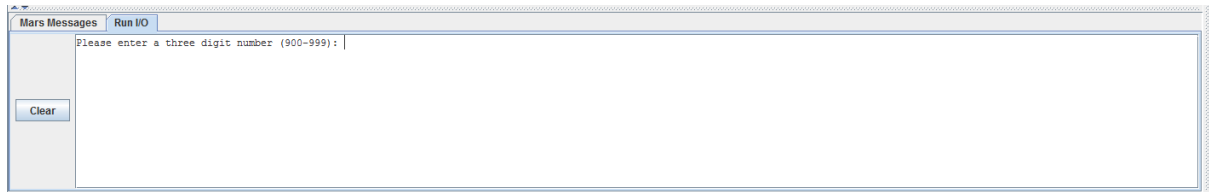
```
#Display ID
li $v0, 1 #Load number
move $a0, $t0 #Moves input into display index
syscall #Shows
```

```
#Display Text
li $v0, 4 #Loads a string type
la $a0, timeElapsed
syscall
```

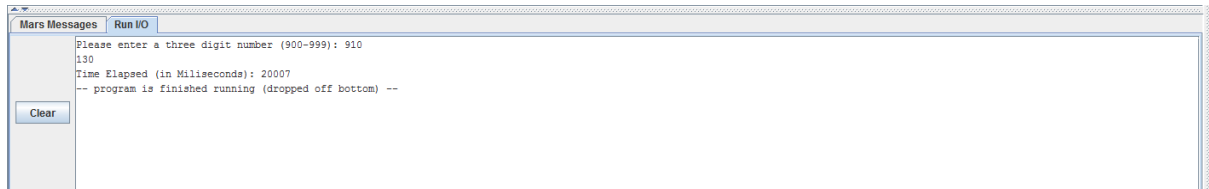
```
#Display Time
li $v0, 1 #Load number
sub $a0, $t7, $t6
syscall #Shows
```

## Images

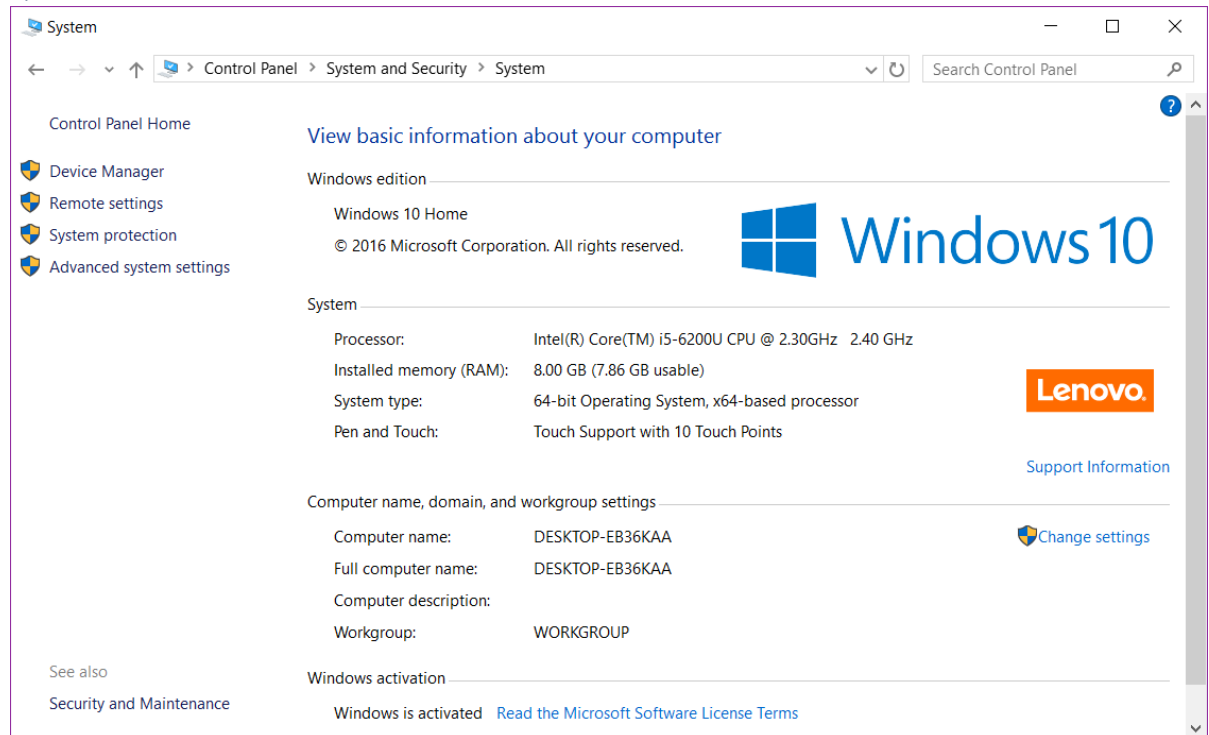
### Input



### Output



## System



## Flow Diagram

