Marmara University

Engineering Faculty

Computer Engineering

Cse 3038 Computer Organization

Project 1 Report

Prof. Haluk Rahmi Topçuoğlu

Group Members

Ahsen Yağmur Kahyaoğlu	150119788
Elif Nur Kemiksiz	100217006
Muhammet Eren Atala	150119904
Nesrin Şimşek	150119664

Before start:

Program clears every registers and buffers because of old variables can be problem.

```
menu:   # At start program clears every registers and buffers
        add $v0,$zero,$zero
        add $v1,$zero,$zero

        add $t0,$zero,$zero
        add $t1,$zero,$zero
        add $t2,$zero,$zero
        add $t3,$zero,$zero
        add $t4,$zero,$zero
        add $t5,$zero,$zero
        add $t6,$zero,$zero
        add $t7,$zero,$zero

        add $s0,$zero,$zero
        add $s1,$zero,$zero
        add $s2,$zero,$zero
        add $s3,$zero,$zero
        # Clears every allocated plase in memory for inputSpace
        la $a1, inputSpace
        clearLoop:
                add $t1,$t0,$a1                 # Gets t0'th location into t1
                sb $zero,0($t1)                 # Stores byte zero into t0£th location

                beq $t0,255,exitClearLoop       # If t0 is in end point, loop will stop
                addi $t0,$t0,1                  # If is not, t0 increaments by one and to be continued
                j clearLoop
        exitClearLoop:
        # Clears every allocated plase in memory for inputSpace2
        la $a1, inputSpace2
        clearLoop2:
                add $t1,$t2,$a1                 # Gets t0'th location into t1
                sb $zero,0($t1)                 # Stores byte zero into t0£th location

                beq $t2,255,exitClearLoop2      # If t0 is in end point, loop will stop
                addi $t2,$t2,1                  # If is not, t0 increaments by one and to be continued
                j clearLoop2
        exitClearLoop2:
```

Program prints the menu, gets the entered choice and executes the related code segment.

```
        exitClearLoop2:
        # Prints menu and gets input from user and starts the related choice
        li $v0,1
        li $v0, 4
        la $a0, menuText
        syscall                                 # Prints menu
        li $v0, 5
        syscall                                 # Gets input from user
        move $t0, $v0
        # Starts the related choice
        beq $t0, 1, baseConverter
        beq $t0, 2, addRationalNumber
        beq $t0, 3, textParser
        beq $t0, 4, mysteryMatrix
        beq $t0, 5, exit
```

Question 1: Program gets 2 input. First input is binary string and second input is type of print solution. In type one, program converts the two's complement binary value into decimal value. In type two, program converts the binary value into hexadecimal value.

```
baseConverter:
    li   $v0, 4                             # Syscall to print prompt string
    la   $a0, enterInput                    # li and la are pseudo instr.
    syscall
    li   $v0, 8                             # Syscall to read input
    la   $a0, inputSpace                    # Allocate space for string input
    li   $a1, 256
    move $t4, $a0
    syscall
    li   $v0, 4                             # Syscall to print prompt string
    la   $a0, enterType                     # li and la are pseudo instr.
    syscall
    li   $v0, 5                             # Syscall to read type
    la   $a0, 0
    syscall
    add $t1, $v0, $zero                     # type stored in $t1
    li   $v0, 1
```

Prints texts and takes input and type from the user.

```
    #  Finds the length of the entered string
    add  $t2, $zero, $zero                   # initial index of binary string
    lenLoop:
        add $t3, $t2, $t4                    # Equals t3 to t4's t2'nd index
        lbu $t3, 0($t3)                      # value of $t3'rd index
        beq $t3, 0, exitLenLoop              # If value inside index equal to
                                             # null char then loop will stop
        addi $t2, $t2, 1
        j lenLoop
    exitLenLoop:
```

Finds length of the input string.

```
    addi $t2, $t2, -1                        # length of binary string
    beq $t1, 1, type1                        # Send to type1 part if t1 is 1
    beq $t1, 2, type2                        # Send to type2 part if t1 is 2
```

Jumps related code segment according to entered type.

```
type1:
    addi $t2, $t2, -1                    # Greatest index in binary string
    add  $t3, $t3, $zero                 # Initial index for loop
    calcLoop:
        add $t5, $t3, $t4
        lbu $t5, 0($t5)                  # value of $t5'th index
        beq $t5, 48, equalZero           # if equal 0 go to equalZero
            addi $t6, $zero, 1           # Equals t6 to 1 for sll operation, if index is equal to 1
            sllv $t6 , $t6, $t2
        beq $t1, 2, notEqualZero
        bne $t3,  $zero, notEqualZero
        sub $t6, $zero, $t6
        notEqualZero:                    # If the index not equal to zero than t7 will increamented by t6
            add $t7, $t6, $t7
            equalZero:
                addi  $t3, $t3, 1        # Increaments by 1 for counter
                addi $t2, $t2, -1        # Discreament by 1 for sll operation
                beq $t2, -1, exitCalcLoop # If all indexes will searched then ends loop
                j calcLoop
                exitCalcLoop:
                    beq $t1, 2, continue # If this part worked for type2 than it will return to related part
                    li   $v0, 4          # Syscall to print prompt string
                    la   $a0, output     # li and la are pseudo instr.
                    syscall              # Prints the Output string
                    move $a0, $t7
                    li $v0, 1
                    syscall              # Prints the calculation
                    j menu
```

Calculates decimal value of binary string.

```
type2:
    li   $v0, 4                          # Syscall to print prompt string
    la   $a0, output                     # li and la are pseudo instr.
    syscall
    add $s0, $zero, $t2                  # Stores length into s0
    div $t2, $t2, 4                      # Splits length
    mfhi $t2                             # Gets the remainder
    add $t0, $t2, $zero                  # Equals t0 to t2

    bne $t2, $zero, notZero              # If there are some numbers  remainder in
                                         # t2 than convert will start in type1 loop
        addi $t2, $zero, 4               # If there is not any before 4 char parts less than 4,
                                         # then t2 equals to 4 and also t0 and sends to type1 loop
        addi $t0, $zero, 4
    notZero:
    j type1

    continue:                            # Finishe the tpe1 loop and prints the sollution
        blt $t7, 10, notConvert          # If number is bigger than or equal to 10 then its
                                         # will be a numerical char
        addi $t7, $t7, 7                 # If number is bigger than or equal to 10 then its
                                         # will be a alphabetic char
        notConvert:
        addi $t7,$t7,48                  # Turns to ascii code
        li $v0, 11
        move $a0, $t7                    # Prints the value
        syscall

        beq $s0, $t0, exitContinue       # If all numbers converted, then loop will stop
        add $t7, $zero, $zero            # If it not, then all loop starts again
                                         # but t7 should be zero for new value enters
        addi $t2, $zero, 3               # For next 4 part, equals t2 to 3
        addi $t0, $t0, 4                 # New lenght will be 4
        j calcLoop
```

Calculates hexadecimal value of binary string.

```
exitContinue:
j menu
```

Jumps menu at the end of first question.

Output:

```
Welcome to our MIPS project!
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option: 1
Input: 10101010
Type: 1
Output: -86
```

```
Type: 1
Output: -86
Welcome to our MIPS project!
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option: 1
Input: 1010101
Type: 2
Output: 55
```

Question 2: Program prints the sum of two rational numbers (fractions) as a rational number too.

```
addRationalNumber:
    li   $v0, 4                          # Syscall to print prompt string
    la   $a0, fNumerator                 # li and la are pseudo instr.
    syscall
    li   $v0, 5                          # Syscall to read first numerator
    li   $a0, 0
    syscall
    add $s0, $v0, $zero                  # type stored in $s0

    li   $v0, 4                          # Syscall to print prompt string
    la   $a0, fDenominator               # li and la are pseudo instr.
    syscall
    li   $v0, 5                          # Syscall to read first denominator
    li   $a0, 0
    syscall
    add $s1, $v0, $zero                  # type stored in $s1

    li   $v0, 4                          # Syscall to print prompt string
    la   $a0, sNumerator                 # li and la are pseudo instr.
    syscall
    li   $v0, 5                          # Syscall to read  second numerator
    li   $a0, 0
    syscall
    add $s2, $v0, $zero                  # type stored in $s2

    li   $v0, 4                          # Syscall to print prompt string
    la   $a0, sDenominator               # li and la are pseudo instr.
    syscall
    li   $v0, 5                          # Syscall to read second denominator
    li   $a0, 0
    syscall
    add $s3, $v0, $zero                  # type stored in $s3
    li   $v0, 1
```

Prints related texts and takes numerator and denominator inputs from the user.

```
    # s0/s1 + s2/s3

    mul $t4, $s0, $s3                    # t4 = s0 * s3
    mul $t5, $s1, $s2                    # t5 = s1 * s2
    mul $t7, $s1, $s3                    # t7 = s1 * s3
    add $t6, $t4, $t5                    # t6 = t4 + t3
```

Does calculations to reach the result.

```
#Euclid
move $t0, $t6
move $t1, $t7
divEuclid:
div $t0, $t1                         # Divides new numerator to denominator
mfhi $t2                             # Gets the remainder into t2
mflo $t3                             # Gets the quotient into t3
beq $t2, 0, stopEuclid               # if remainder equals to zero, then loop will stop
move $t0, $t1                        # if it's not then old denominator will be new numerator
move $t1, $t2                        # And remainder will be new denominator
j divEuclid
stopEuclid:
# $t1 is the gcc
# Divedes numerator and denominator to gcc
```

Calculates remainder and quotient until the remainder is zero.

```
    # Divedes numerator and denominator to gcc
    div $t2, $t6, $t1
    div $t3, $t7, $t1
```

Calculates numerator and denominator of the actual result.

```
    # Prints the all equation
    li   $v0, 4                          # Syscall to print prompt string
    la   $a0, output2                    # li and la are pseudo instr.
    syscall
```

Prints “Output:”

Output:

```
Welcome to our MIPS project!
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option: 2
Enter the first numerator: 1
Enter the first denominator: 2
Enter the second numerator: 1
Enter the second denominator: 3
Output:
1 / 2 + 1 / 3 = 5 / 6
```

Question 3: Program takes an input sentence and prints the words in it by parsing with given set of characters. A word is a sequence of characters without any whitespaces and given set of punctuation marks. The only punctuation marks that you have to consider will be given as second input

```
textParser:
li   $v0, 4                          # Syscall to print prompt string
la   $a0, enterInput                 # li and la are pseudo instr.
syscall
li   $v0, 4                          # Syscall to print prompt string
la   $a0, enterInputText             # li and la are pseudo instr.
syscall
li   $v0, 8                          # Syscall to read text
la   $a0, inputSpace
li   $a1, 256
move $t0, $a0                        #t0 holds text's start address
syscall

li   $v0, 4                          # Syscall to print prompt string
la   $a0, parserChars                # li and la are pseudo instr.
syscall
li   $v0, 8                          # Syscall to read parser characters
la   $a0, inputSpace2
li   $a1, 256
move $t1, $a0                        # t1 holds parser characters' start address
syscall
```

Prints related texts and takes input text and parser characters from the user.

```
 li   $v0, 4                              # Syscall to print prompt string
 la   $a0, output2                        # li and la are pseudo instr.
 syscall
```

Prints “Output:”

```
add  $t2, $zero, $zero               # initial index of text is zero at first
textLoop:
    add  $t4, $zero, $zero           # initial index of parser chars is zero at first
    add $t3, $t2, $t0
    lbu $t3, 0($t3)                  # value of $t3'rd index
    beq $t3, 10, exitTextLoop        # exits text loop if it is end of the text
    bne $t3, 32, parserLoop          # jumps putLineFeed if the char is space
    j putLineFeed
    continueTextLoop:
    li   $v0, 11
    move $a0, $t3
    syscall                          # system call to print char
    addi $t2, $t2, 1                 # increments index to search text
    j textLoop
exitTextLoop:
    j menu
parserLoop:
    add $t5, $t4, $t1
    lbu $t5, 0($t5)                  # value of $t5'th index of parser characters
    beq $t5, 10, exitParserLoop      # exits parser loop if it is end of the parser character is
    beq $t5, $t3, putLineFeed        # jumps putLineFeed if text char and parser char is matched
    addi $t4, $t4, 1                 # increments index to search parser characters
    j parserLoop
    putLineFeed:
        addi $t6, $zero, 10
        add $t3, $t2, $t0
        sb $t6, 0($t3)               # stores \n character in related index of text
        lbu $t3, 0($t3)              # value of $t3'rd index
        j continueTextLoop
exitParserLoop:
    j continueTextLoop
```

Searchs input text and prints related character.

Searchs parser characters and if it is matched with related character of input text, assigns “\n” to that character.

Output:

```
Welcome to our MIPS project!
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option: 3
Input:
Input text: De!n?em%e
Parser characters: !?%
Output:
De
n
em
e
```

Question 4: takes an input sentence and prints the words in it by parsing with given set of characters. A word is a sequence of characters without any whitespaces and given set of punctuation marks. The only punctuation marks that you have to consider will be given as second input.

```
mysteryMatrix:
    li   $v0, 4                                 # Syscall to print prompt string
    la   $a0, enterInput                        # li and la are pseudo instr.
    syscall
    # gets input
    li   $v0, 8                                 # Syscall to read array
    la   $a0, inputSpace                        # a0 = address of inputSpace
    li   $a1, 256
    move $t0, $a0                               # t0 = array's start address
    syscall
    li   $v0, 4                                 # Syscall to print prompt string
    la   $a0, output2                           # li and la are pseudo instr.
    syscall
```

Prints related text and takes matrix input from the user.

```
    li   $v0, 4                                 # Syscall to print prompt string
    la   $a0, output2                           # li and la are pseudo instr.
    syscall
```

Prints “Output:”

```
    add $t4, $zero, $zero                       # array counter
    add $t2, $zero, $zero                       # Counter = 0
    add $t1, $zero, $zero                       # Sum = 0
    convertToIntLoop:
        add $t3, $t0, $t2
        lbu $t3, 0($t3)                         # t3 = value of array
        beq $t3, 32, whitespace                 # jumps whitespace if the char is a whitespace (means end of the number)
        beq $t3, 10, whitespace                 # jumps whitespace if the char is \n (means end of the last number)
        mul $t1, $t1, 10                        # multiplies the number with 10 (means searching the number has not finished yet)
        addi $t3, $t3, -48                      # converts char to integer
        add $t1, $t1, $t3                       # sums to find actual value of number
        addi $t2, $t2, 1                        # increments index to search array
        j convertToIntLoop
    whitespace:
        sll $t5, $t4, 2                         # multiplies the index with four to find the address
        addi $t2, $t2, 1                        # increments index to search array
        la $a0, inputSpace2                     # a0 = address of inputSpace2
        move $t6, $a0                           # t6 = integer array
        add $t7, $t6, $t5
        sw $t1, 0($t7)                          # stores integer value of array at address t7
        add $t1, $zero, $zero                   # t1 = zero at first for each number
        addi $t4, $t4, 1                        # increments t4 to keep the size of the array
        beq $t3, 10, exitConvertToIntLoop       # jumps exitConvertToIntLoop if the char is \n (means end of the last number)
        j convertToIntLoop
    exitConvertToIntLoop:
```

Converts chars in the input matrix to integers and calculates the integer numbers.

Creates an array and stores these numbers in this array.

```
    add $t3, $zero, $zero
    add $t5, $zero, $zero

    sqrt:
        addi $t3, $t3, 2                        # increments t3 by two because size of matrix must be even
        mul $t5, $t3, $t3                       # calculates square of t3
        beq $t4, $t5, endSqrt                   # exits loop (means squareroot of size of the array (N) is found)
        j sqrt
    endSqrt:
```

Calculates squareroot of size of the matrix.

```
    mul $t2, $t3, 2                             # t2 = 2N
    add $t7, $t2, $zero                         # t7 = 2N
    addi $t2, $t2, -1                           # t2 = last index for left to right (2N - 1)
    addi $t3, $t3, -1                           # t3 = last index of first row (N - 1)
    addi $t4, $t4, -1                           # t4 = last index of input array (NxN) - 1
    addi $t5, $zero, 1                          # multiplication = 1 at first

    leftToRight:
        add $t1, $t6, $t1                       # t1 = address of t1'st index of integer array
        lw $t1, 0($t1)                          # t1 = value of t1'st index of integer array
        mul $t5, $t1, $t5                       # t5 = multiplication result
        beq $t0, $t2, printMultiplication       # jumps printMultiplication if index = 2N-1 (means end of first two row)
        addi $t0, $t0, 1                        # increments index by one
        bne $t0, $t3, onTheSameRow              # jumps onTheSameRow if index is not equal to N-1
        addi $t0, $t0, 1                        # increments index if index is equal to N-1 (extra increment)

    onTheSameRow:
        addi $t0, $t0, 1                        # increments index by one
        sll $t1, $t0, 2                         # multiplies the index with four to find the address
        ble $t0, $t2, leftToRight               # continues if index <= 2N-1 (means it is not end of the array yet)

    printMultiplication:
        move $a0, $t5
        li $v0, 1
        syscall                                 # system call to print multiplication
        addi $t5, $zero, 32                     # t5 = 32 (ascii code of space)
        move $a0, $t5
        li $v0, 11
        syscall                                 # system call to print space

        addi $t5, $zero, 1                      # multiplication = 1 again
        add $t3, $t3, $t7                       # t3 = t3 + 2N
        add $t2, $t2, $t7                       # t2 = t2 + 2N
        blt $t0, $t4, onTheSameRow              # continue if it is not end of the array
        j menu
```

Calculates last index of array, squareroot of array and first row of the array.

Multiplies numbers on related index from left to right according to homework explanation.

Prints multiplications.

Output:

```
Welcome to our MIPS project!
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option: 4
Input: 3 8 12 1 2 3 4 3 2 4 5 4 5 6 7 11 4 2 9 3 23 14 5 58 5 3 4 5 7 8 4 2 4 9 1 2
Output:
3456 341040 5040
```

Question 5: Exits

```
434     exit:
435         li   $v0, 4                         # Syscall to print prompt str
436         la   $a0, programEnds               # li and la are pseudo instr.
437         syscall
438         li   $v0, 10                        # Syscall to exit
439         syscall
```

Prints a message that indicates program terminated.

Output:

```
Welcome to our MIPS project!
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option: 5
Program ends. Bye :)
-- program is finished running --
```