Meng Wai Chan
Mar 14, 2023

RATIONAL NUMBER ARITHMETIC ASSIGNMENT

Meng Wai Chan
Professor Gertner
CSc 342
March 12, 2023

Meng Wai Chan
Mar 14, 2023

Table of Contents

Contents

Meng Wai Chan
Mar 14, 2023

Objective:

The goal of this assignment was to learn how to do rational number arithmetic without using any floating point numbers, Furthermore the objective is to investigate how MIPS assembly and C++ can perform those arithmetic and observe how they are performed.

Description:

In C++ we created a Class called Rational that will take 2 inputs p and q to construct the rational number. **See Figure 1.** Then we have a 2 function call is_rationa() to check if the number given is a rational number by checking if q = 0, if q is equal to 0 the program will let the user know the number given is not valid. **See Figure 1.** Next we create a print_rational() to output the rational number given by the user if it is valid. **Figure 1.** Next part of the task is to perform arithmetic of adding **(Figure 2),** subtracting **(Figure 3),** multiplying **(Figure 4),** and dividing **(Figure 5),** by overloading the operators +, -, *, / by doing this the user is able to perform arithmetic using r1 + r2 to find the sum of the rational numbers. In the following pages you are able to see how they are performed and the output for each function. **(Figure 7)**

In MIPS Assembly we have to recreate the add**(Page 13)**, subtract **(Page 14),** multiply **(Page 15),** divide **(Page 16)** in MARS by performing the arithmetic using the address in the assembly and output the answers into the register.

Meng Wai Chan
Mar 14, 2023

Codes for Rational Number Arithmetic In C++ (Task 1)

```cpp
#include <iostream>
using namespace std;

class Rational{
    public:
    int p;
    int q;

    Rational(int x = 0, int y = 1){
        p = x;
        q = y;
    }

    bool is_rational(){
        if(this->q == 0){
            return false;
        }

        return true;
    }

    void print_rational(){
        if(this->is_rational()){
            cout<<"Rational("<<this->p<<"/"<<this->q<<")";
        }
        else{
            cout<<"Invalid Rational Number";
        }
    }
};
```

Figure 1. Rational Class, is_rationa(), print_rational()

```
32  Rational operator + (Rational& r1, Rational& r2){
33      if(!r1.is_rational() || !r2.is_rational()){
34          Rational temp(0,0);
35          return temp;
36      }
37      int p, q;
38      if(r1.q != r2.q){
39          p = r1.p*r2.q + r2.p*r1.q;
40          q = r1.q*r2.q;
41      }
42      else{
43          p = r1.p + r2.p;
44          q = r1.q;
45      }
46
47      Rational result(p,q);
48
49      return result;
50  }
51
```

Figure 2. add_Rational()

```
52  Rational operator - (Rational& r1, Rational& r2){
53      if(!r1.is_rational() || !r2.is_rational()){
54          Rational temp(0,0);
55          return temp;
56      }
57
58      int p, q;
59      if(r1.q != r2.q){
60          p = r1.p*r2.q - r2.p*r1.q;
61          q = r1.q*r2.q;
62      }
63      else{
64          p = r1.p - r2.p;
65          q = r1.q;
66      }
67
68      Rational result(p,q);
69      return result;
70  }
```

Figure 3. sub_Rational()

```
72   Rational operator * (Rational& r1, Rational& r2){
73       if(!r1.is_rational() || !r2.is_rational()){
74           Rational temp(0,0);
75           return temp;
76       }
77
78       int p = r1.p * r2.p;
79       int q = r1.q * r2.q;
80       Rational result(p, q);
81
82       return result;
83   }
84
```

Figure 4. Mul_rational()

```
85  Rational operator / (Rational& r1, Rational& r2){
86      Rational temp(0,0);
87      if(!r1.is_rational() || !r2.is_rational()){
88          return temp;
89      }
90      int p = r1.p * r2.q;
91      int q = r1.q * r2.p;
92
93      Rational result(p, q);
94      if(!result.is_rational()){
95          return temp;
96      }
97
98      return result;
99  }
100
```

Figure 5. Div_rational()

```
101 void print_arithmetic(Rational r1, Rational r2, Rational r3, string s){
102     r1.print_rational();
103     cout<<s;
104     r2.print_rational();
105     cout<<" = ";
106     r3.print_rational();
107     cout<<"\n";
108 }
109
```

Figure 6

```
110  int main(){
111      int a, b, c, d;
112
113      cout<<"Rational(a, b) :\nEnter a = ";
114      cin>>a;
115      cout<<"Enter b = ";
116      cin>>b;
117      cout<<"Rational(c, d) :\nEnter c = ";
118      cin>>c;
119      cout<<"Enter d = ";
120      cin>>d;
121
122      Rational r1(a,b);
123      Rational r2(c,d);
124
125      Rational r3 = r1 + r2;
126      Rational r4 = r1 - r2;
127      Rational r5 = r1 * r2;
128      Rational r6 = r1 / r2;
129
130      r1.print_rational(); cout<<endl;
131
132      r2.print_rational(); cout<<endl;
133
134      print_arithmetic(r1, r2, r3, " + ");
135      print_arithmetic(r1, r2, r4, " - ");
136      print_arithmetic(r1, r2, r5, " * ");
137      print_arithmetic(r1, r2, r6, " / ");
138
139      return 0;
140  }
141
```

Figure 7, (Task 2) main()

Meng Wai Chan
Mar 14, 2023

Outputs for Rational Number Arithmetic

```
Rational(a, b) :
Enter a = 1
Enter b = 2
Rational(c, d) :
Enter c = 3
Enter d = 4
Rational(1/2)
Rational(3/4)
Rational(1/2) + Rational(3/4) = Rational(10/8)
Rational(1/2) - Rational(3/4) = Rational(-2/8)
Rational(1/2) * Rational(3/4) = Rational(3/8)
Rational(1/2) / Rational(3/4) = Rational(4/6)
```

Figure 8, Output 1

```
Rational(a, b) :
Enter a = 1
Enter b = 2
Rational(c, d) :
Enter c = 0
Enter d = 1
Rational(1/2)
Rational(0/1)
Rational(1/2) + Rational(0/1) = Rational(1/2)
Rational(1/2) - Rational(0/1) = Rational(1/2)
Rational(1/2) * Rational(0/1) = Rational(0/2)
Rational(1/2) / Rational(0/1) = Invalid Rational Number
```

Figure 9, Output 2

```
Rational(a, b) :
Enter a = 0
Enter b = 1
Rational(c, d) :
Enter c = 1
Enter d = 0
Rational(0/1)
Invalid Rational Number
Rational(0/1) + Invalid Rational Number = Invalid Rational Number
Rational(0/1) - Invalid Rational Number = Invalid Rational Number
Rational(0/1) * Invalid Rational Number = Invalid Rational Number
Rational(0/1) / Invalid Rational Number = Invalid Rational Number
```

Figure 10, Output 3

Meng Wai Chan
Mar 14, 2023

Codes for Rational Number Arithmetic In MIPS Assembly (Task 3)

```
.data
num1:
        .word 1
        .word 2
num2:
        .word 1
        .word 4
result:
        .word 0
        .word 0

.text
main:
        la $t0, num1    # load address of num1 into $t0
        lw $t1, 0($t0)  # load p of num1 into $t1
        lw $t2, 4($t0)  # load q of num1 into $t2

        la $t0, num2    # load address of num2 into $t0
        lw $t3, 0($t0)  # load p of num2 into $t3
        lw $t4, 4($t0)  # load q of num3 into $t4

        # multiply
        mul $t1, $t1, $t4       # $t1 = $t1 * $t4
        mul $t3, $t3, $t2       # $t3 = $t3 * $t2

        # add num1.p and num2.p
        add $t1, $t1, $t3       # $t1 = $t1 + $t3

        # multiply num1.q and num2.q
        mul $t2, $t2, $t4       # $t2 = $t2 * $t4

        # store result
        la $t0, result  # load address of result into #t0
        sw $t1, 0($t0)  # store p to result
        sw $t2, 4($t0)  # store q to result

        li $v0, 10
        syscall
```

$t1	9	0x00000006
$t2	10	0x00000008

```
.data
num1:
        .word 3
        .word 4
num2:
        .word 5
        .word 6
result:
        .word 0
        .word 0

.text
main:
        la $t0, num1    # load address of num1 into $t0
        lw $t1, 0($t0)  # load p of num1 into $t1
        lw $t2, 4($t0)  # load q of num1 into $t2

        la $t0, num2    # load address of num2 into $t0
        lw $t3, 0($t0)  # load p of num2 into $t3
        lw $t4, 4($t0)  # load q of num3 into $t4

        # multiply num1.p and num2.q
        # multiply num2.p and num1.q
        mul $t1, $t1, $t4       # $t1 = $t1 * $t4
        mul $t3, $t3, $t2       # $t3 = $t3 * $t2

        # add num1.p and num2.p
        sub $t1, $t1, $t3       # $t1 = $t1 - $t3

        # multiply num1.q and num2.q
        mul $t2, $t2, $t4       # $t2 = $t2 * $t4

        # store result
        la $t0, result  # load address of result into #t0
        sw $t1, 0($t0)  # store p to result
        sw $t2, 4($t0)  # store q to result

        li $v0, 10
        syscall
```

$t1	9	0xfffffffe
$t2	10	0x00000018

```
1  .data
2  num1:
3          .word 3
4          .word 4
5  num2:
6          .word 5
7          .word 6
8  result:
9          .word 0
10         .word 0
11
12 .text
13 main:
14         la $t0, num1    # load address of num1 into $t0
15         lw $t1, 0($t0)  # load p of num1 into $t1
16         lw $t2, 4($t0)  # load q of num1 into $t2
17
18         la $t0, num2    # load address of num2 into $t0
19         lw $t3, 0($t0)  # load p of num2 into $t3
20         lw $t4, 4($t0)  # load q of num3 into $t4
21
22         # multiply num1.p and num2.p
23         mul $t1, $t1, $t3       # $t1 = $t1 * $t3
24
25         # multiply num1.q and num2.q
26         mul $t2, $t2, $t4       # $t2 = $t2 * $t4
```

```
28         # store result
29         la $t0, result  # load address of result into #t0
30         sw $t1, 0($t0)  # store p to result
31         sw $t2, 4($t0)  # store q to result
32
33         li $v0, 10
34         syscall
```

$t1	9	0x0000000f
$t2	10	0x00000018

```
1  .data
2  num1:
3          .word 3
4          .word 4
5  num2:
6          .word 5
7          .word 6
8  result:
9          .word 0
10         .word 0
11
12 .text
13 main:
14         la $t0, num1    # load address of num1 into $t0
15         lw $t1, 0($t0)  # load p of num1 into $t1
16         lw $t2, 4($t0)  # load q of num1 into $t2
17
18         la $t0, num2    # load address of num2 into $t0
19         lw $t3, 0($t0)  # load p of num2 into $t3
20         lw $t4, 4($t0)  # load q of num3 into $t4
21
22         # multiply num1.p and num2.q
23         mul $t1, $t1, $t4       # $t1 = $t1 * $t4
24
25         # multiply num1.q and num2.p
26         mul $t2, $t2, $t3       # $t2 = $t2 * $t3
```

```
28         # store result
29         la $t0, result  # load address of result into #t0
30         sw $t1, 0($t0)  # store p to result
31         sw $t2, 4($t0)  # store q to result
32
33         li $v0, 10
34         syscall
```

$t1	9	0x00000012
$t2	10	0x00000014

Meng Wai Chan
Mar 14, 2023

Task 4

```
.macro print_rational(%p, %q)
        .data
                str_rational: .asciiz "rational("
                slash: .asciiz "/"
                newLine: .asciiz ")\n"
        .text
        li $v0, 4
        la $a0, str_rational
        syscall
        li $v0, 1
        move $a0, %p
        syscall

        li $v0, 4
        la $a0, slash
        syscall

        li $v0, 1
        move $a0, %q
        syscall

        li $v0, 4
        la $a0, newLine
        syscall
.end_macro
```

```
.macro is_rational(%p, %q)
        .data
                str_notRational: .asciiz "Invalid Rational Number"
        .text
        move $t0, %q
        beq $t0, $zero, invalid
        li $v0, 10
        syscall

        invalid:
                li $v0, 4
                la $a0, str_notRational
                syscall
.end_macro
```

```
42  .macro add_rational(%p1, %q1, %p2, %q2)
43          .text
44                  mult %p1, %q2
45                  mflo $t0
46                  mult %p2, %q1
47                  mflo $t1
48                  mult %q1, %q2
49                  mflo $t2
50
51                  add $t0, $t0, $t1
52
53                  print_rational($t0, $t2)
54
55                  li $v0, 10
56  .end_macro
```

```
58  .macro sub_rational(%p1, %q1, %p2, %q2)
59          .text
60                  mult %p1, %q2
61                  mflo $t0
62                  mult %p2, %q1
63                  mflo $t1
64                  mult %q1, %q2
65                  mflo $t2
66
67                  sub $t0, $t0, $t1
68
69                  print_rational($t0, $t2)
70
71                  li $v0, 10
72  .end_macro
```

```
74  .macro mul_rational(%p1, %q1, %p2, %q2)
75          .text
76                  mult %p1, %p2
77                  mflo $t0
78                  mult %q1, %q2
79                  mflo $t1
80
81                  print_rational($t0, $t1)
82
83                  li $v0, 10
84  .end_macro
86  .macro div_rational(%p1, %q1, %p2, %q2)
87          .text
88                  mult %p1, %q2
89                  mflo $t0
90                  mult %q1, %p2
91                  mflo $t1
92
93                  print_rational($t0, $t1)
94
95                  li $v0, 10
96  .end_macro
```

```
 98  .text
 99  main:
100          addi $s0, $0, 1
101          addi $s1, $0, 2
102          addi $s2, $0, 1
103          addi $s3, $0, 4
104
105          add_rational($s0,$s1,$s2,$s3)
106          sub_rational($s0,$s1,$s2,$s3)
107          mul_rational($s0,$s1,$s2,$s3)
108          div_rational($s0,$s1,$s2,$s3)
109
110          print_rational($s0, $s1)
111
112          addi $s4, $0, 1
113          addi $s5, $0, 0
114          is_rational($s4, $s5)
115
116          li $v0, 10
117          syscall
```

```
rational(6/8)
rational(2/8)
rational(1/8)
rational(4/2)
rational(1/2)
Invalid Rational Number
```

Conclusion

In this assignment I have a deeper understanding of how rational numbers are added in assembly and C++, by creating add_rational(), sub_rational(), mul_rational(), div_rational(), is_rational(), print_rational(). By recreating the same function in MIPS assembly using macros I understand how to create them in MIPS assembly, and am able to witness how each operation is called and function.