The ISA simulator

GaoYang 6019545

CONTENTS


01
Predefined

Predefined

Operator	4-bit	Operands	3-bit	17-bit
mov	0001	r0	000	00000000000000000
add	0010	r1	001	00000000000000000
sub	0011	r2	010	00000000000000000
mul	0100	r3	011	00000000000000000
div	0101	r4	100	00000000000000000
jmp	0110	r5	101	00000000000000000
mem	0111	r6	110	00000000000000000
end	0000	r7	111	00000000000000000
24-bit				

02 Calculate the core

Calculate the core

```
//真正的核心函数
private static string cul2(string[] operand1, string[] values1, string[] values2, int[] intvalues, string s, string[,] codes, strin
{
    int nus = 0;
    for (int i = 0; i < codes.GetLength(0); i++)
    {
        if (codes[i, 0] == "mov")...
        if (codes[i, 0] == "add")...
        if (codes[i, 0] == "sub")...
        if (codes[i, 0] == "mul")...
        if (codes[i, 0] == "div")...
        if (codes[i, 0] == "jmp")...
    }
    return s;
}
```

```
323     private static string cul2(string[] operand1, string[] values1, string[] values2, int[] intvalues, string s, string[,] codes, strin
324     {
325         int nus = 0;
326         for (int i = 0; i < codes.GetLength(0); i++)
327         {
328             if (codes[i, 0] == "mov")//是不是mov指令
329             {
330                 if (System.Text.RegularExpressions.Regex.IsMatch(codes[i, 2], @"^[-]?[0-9]+$"))//正则表达式判断是数字
331                 {
332                     s = codes[i, 2];
333                     int num = Array.IndexOf(operand1, codes[i, 1]);//索引出变量下标
334                     int s0 = Convert.ToInt32(s);//代码的立即数直接转整型
335                     s = ConvertBits(s0).ToString();//整数转二进制字符串
336                     values2[num] = s;//覆盖原始值
337                     nus = nus + 1;
338                     culvalue(values2, intvalues);//补足32位
339                     for (int ix = 0; ix < values1.Length; ix++)//输出计算后的结果
340                     {
341                         if (ix == num)
342                         {
343                             Console.WriteLine("{0} = {1} [{2}]", values1[ix], intvalues[ix], values2[ix]);
344                         }
345                         else
346                         {
347                             // Cons
348                         }
349                     }
350                     continue;
351                 }
352                 else //正则表达式判
```

```
352                 else //正则表达式判断失败 是变量
353                 {
354                     int num = Array.IndexOf(operand1, codes[i, 2]);//索引出第一个变量下标
355                     int num0 = Array.IndexOf(operand1, codes[i, 1]);//索引出第二个变量下标
356                     values2[num0] = values2[num];//用第二个值覆盖第一个值
357                     nus = nus + 1;
358                     culvalue(values2, intvalues);//补齐32位
359                     for (int ix = 0; ix < values1.Length; ix++)//显示
360                     {
361                         if (ix == num)
362                         {
363                             Console.WriteLine("{0}={1} [{2}]", values1[ix], intvalues[ix], values2[ix]);
364                         }
365                         else
366                         {
367                             // Console.WriteLine("{0} : [{1}] : {2}", values1[ix], values2[ix], intvalues[ix]);
368                         }
369                     }
370                     continue;
371                 }
372             }
```

MOV

```
if (codes[i, 0] == "add")
{
    if (System.Text.RegularExpressions.Regex.IsMatch(codes[i, 2], @"^[-]?[0-9]+$"))
    {
        int num = Array.IndexOf(operand1, codes[i, 1]);
        s = codes[i, 2];
        int s0 = Convert.ToInt32(s);
        string stru = values2[num];
        if (stru.Length <= 32)
        {
            for (int sn = stru.Length; sn < 32; sn++)
            {
                if (stru[0] == '1')
                {
                    stru = '1' + stru;
                }
                else
                {
                    stru = '0' + stru;
                }
            }
        }
        int s1 = Convert.ToInt32(stru, 2);
        s1 = s1 + s0;
        s = ConvertBits(s1).ToString();
        values2[num] = s;
        nus = nus + 1;
        culvalue(values2, intvalues);
        culvalue(values2, intvalues);
        for (int ix = 0; ix < values1.Length; ix++)
        {
            if (ix == num)
            {
                Console.WriteLine("{0}={1} [{2}]", values1[ix], intvalues[ix], values2[ix]);
            }
            else
            {
                // Console.WriteLine("{0} : [{1}] : {2}", values1[ix], values2[ix], intvalues
            }
        }
        continue;
    }
    else
    {
        int num1 = Array.IndexOf(operand1, codes[i, 1]);
        int num2 = Array.IndexOf(operand1, codes[i, 2]);
        string str1 = values2[num1];
        string str2 = values2[num2];
        int s1 = Convert.ToInt32(str1, 2);
        int s2 = Convert.ToInt32(str2, 2);
        s1 = s1 + s2;
        s = ConvertBits(s1).ToString();
        values2[num1] = s;
        nus = nus + 1;
        culvalue(values2, intvalues);
        for (int ix = 0; ix < values1.Length; ix++)
        {
```

add

```
if (codes[i, 0] == "sub")
{
    if (System.Text.RegularExpressions.Regex.IsMatch(codes[i, 2], @"^[-]?[0-9]+$"))
    {
        int num = Array.IndexOf(operand1, codes[i, 1]);
        s = codes[i, 2];
        int s0 = Convert.ToInt32(s);
        string stru = values2[num];
        if (stru.Length <= 32)
        {
            for (int sn = stru.Length; sn < 32; sn++)
            {
                if (stru[0] == '1')
                {
                    stru = '1' + stru;
                }
                else
                {
                    stru = '0' + stru;
                }
            }
        }
        int s1 = Convert.ToInt32(stru, 2);
        s1 = s1 - s0;
        s = ConvertBits(s1).ToString();
        values2[num] = s;
        nus = nus + 1;
        culvalue(values2, intvalues);
```

```
    else
    {
        int num1 = Array.IndexOf(operand1, codes[i, 1]);
        int num2 = Array.IndexOf(operand1, codes[i, 2]);
        string str1 = values2[num1];
        string str2 = values2[num2];
        int s1 = Convert.ToInt32(str1, 2);
        int s2 = Convert.ToInt32(str2, 2);
        s1 = s1 - s2;
        s = ConvertBits(s1).ToString();
        values2[num1] = s;
        nus = nus + 1;
        culvalue(values2, intvalues);
        for (int ix = 0; ix < values1.Length; ix++)
        {
            if (ix == num1)
            {
                Console.WriteLine("{0}={1} [{2}]", values1[ix], intvalues[ix], values2[ix]);
            }
            else
            {
                // Console.WriteLine("{0} : [{1}] : {2}", values1[ix], values2[ix], intvalues
            }
        }
        continue;
    }
```

sub

```
if (codes[i, 0] == "mul")
{
    if (System.Text.RegularExpressions.Regex.IsMatch(codes[i, 2], @"^[-]?[0-9]+$"))
    {
        int num = Array.IndexOf(operand1, codes[i, 1]);
        s = codes[i, 2];
        int s0 = Convert.ToInt32(s);
        string stru = values2[num];
        if (stru.Length <= 32)
        {
            for (int sn = stru.Length; sn < 32; sn++)
            {
                if (stru[0] == '1')
                {
                    stru = '1' + stru;
                }
                else
                {
                    stru = '0' + stru;
                }
            }
        }
        int s1 = Convert.ToInt32(stru, 2);
        s1 = s1 * s0;
        s = ConvertBits(s1).ToString();
        string ssr = s;
        if (ssr.Length <= 32)
        {
            for (int sn = ssr.Length; sn < 32; sn++)
            {
```

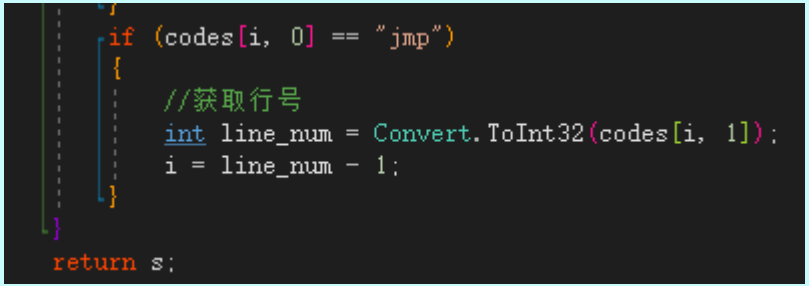
```
else
{
    int num1 = Array.IndexOf(operand1, codes[i, 1]);
    int num2 = Array.IndexOf(operand1, codes[i, 2]);
    string str1 = values2[num1];
    string str2 = values2[num2];
    int s1 = Convert.ToInt32(str1, 2);
    int s2 = Convert.ToInt32(str2, 2);
    s1 = s1 * s2;
    s = ConvertBits(s1).ToString();
    string ssr = s;
    if (ssr.Length <= 32)
    {
        for (int sn = ssr.Length; sn < 32; sn++)
        {
            if (ssr[0] == '1')
            {
                ssr = '1' + ssr;
            }
            else
            {
                ssr = '0' + ssr;
            }
        }
    }
    rm = ssr;
    values2[num1] = s;
    nus = nus + 1;
    culvalue(values2, intvalues);
    for (int ix = 0; ix < values1.Length; ix++)
```

mul

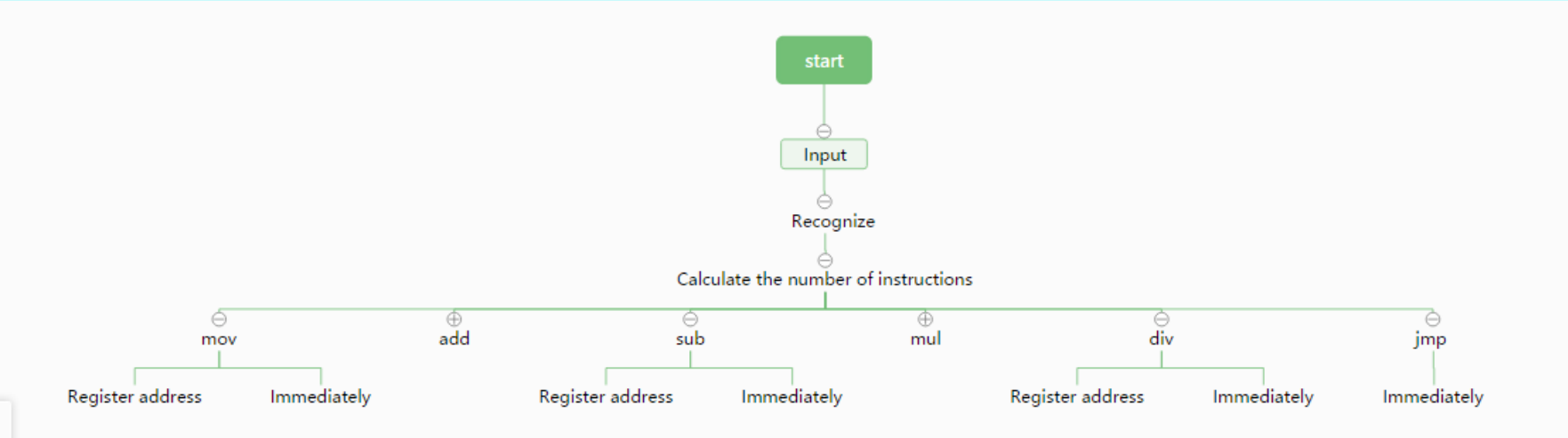
```
if (codes[i, 0] == "div")
{
    if (System.Text.RegularExpressions.Regex.IsMatch(codes[i, 2], @"^[-]?[0-9]+$"))
    {
        int num = Array.IndexOf(operand1, codes[i, 1]);
        s = codes[i, 2];
        int s0 = Convert.ToInt32(s);
        string stru = values2[num];
        if (stru.Length <= 32)
        {
            for (int sn = stru.Length; sn < 32; sn++)
            {
                if (stru[0] == '1')
                {
                    stru = '1' + stru;
                }
                else
                {
                    stru = '0' + stru;
                }
            }
        }
        int s1 = Convert.ToInt32(stru, 2);
        if (s0 == 0)
        {
            Console.WriteLine("0 can not do divisor");
            break;
        }
        int dc = s1 % s0;
```

```
else
{
    int num1 = Array.IndexOf(operand1, codes[i, 1]);
    int num2 = Array.IndexOf(operand1, codes[i, 2]);
    string str1 = values2[num1];
    string str2 = values2[num2];
    int s1 = Convert.ToInt32(str1, 2);
    int s2 = Convert.ToInt32(str2, 2);
    if (s2 == 0)
    {
        Console.WriteLine("0 can not do divisor");
        break;
    }
    int dc = s1 % s2;
    re = ConvertBits(dc).ToString();
    s1 = s1 / s2;
    s = ConvertBits(s1).ToString();
    values2[num1] = s;
    nus = nus + 1;
    culvalue(values2, intvalues);
    for (int ix = 0; ix < values1.Length; ix++)
    {
        if (ix == num1)
        {
            Console.WriteLine("{0}={1} [{2}] re={3} [
        }
        else
        {
            // Console.WriteLine("{0} : [{1}] : {2}", v
```

div


if (codes[i, 0] == "jmp")
{
//获取行号
int line_num = Convert.ToInt32(codes[i, 1]);
i = line_num - 1;
}
}
return s;

jmp

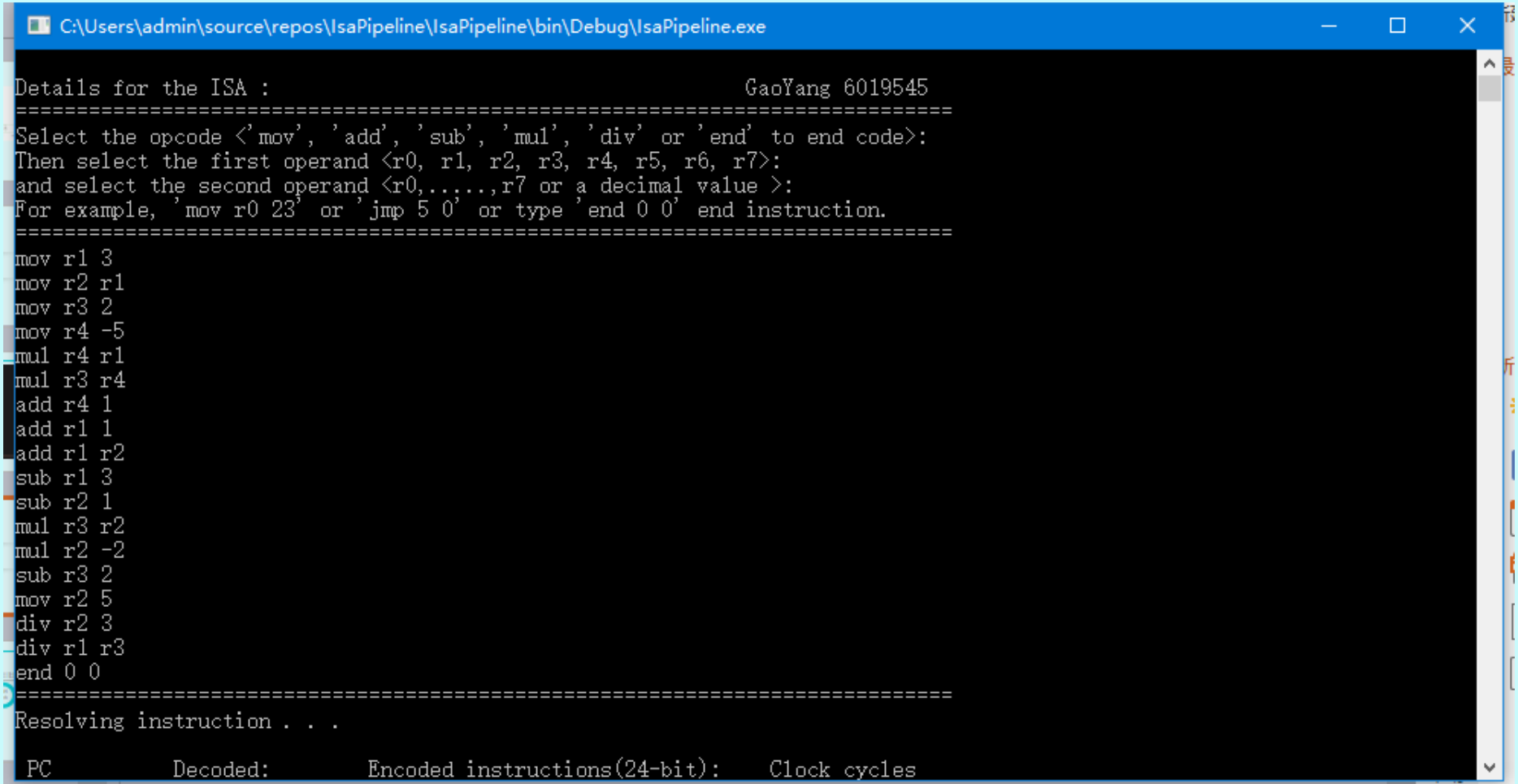

start
Input
Recognize
Calculate the number of instructions
mov
add
sub
mul
div
jmp
Register address
Immediately
Register address
Immediately
Register address
Immediately
Immediately

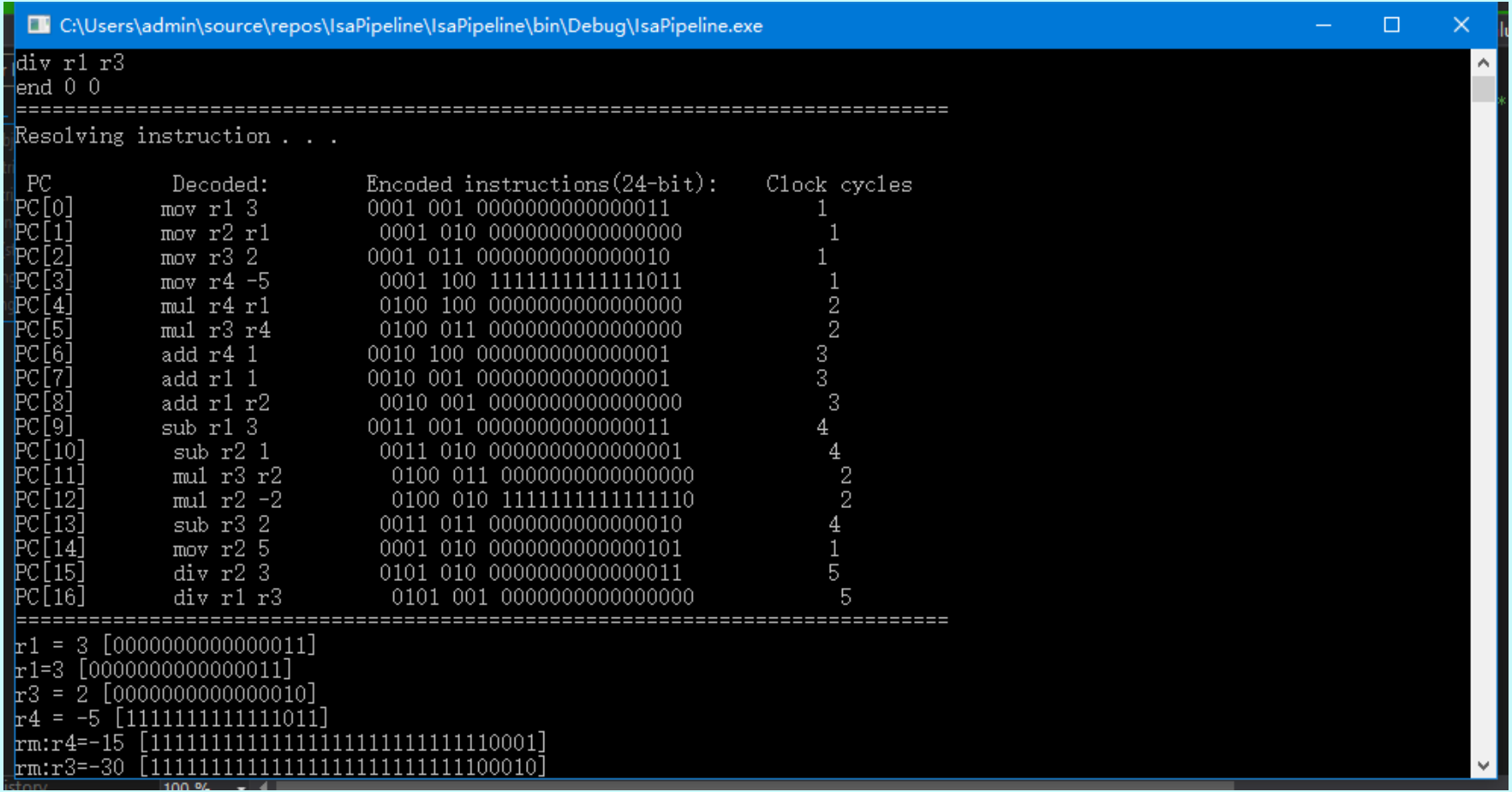
03 Pipeline core

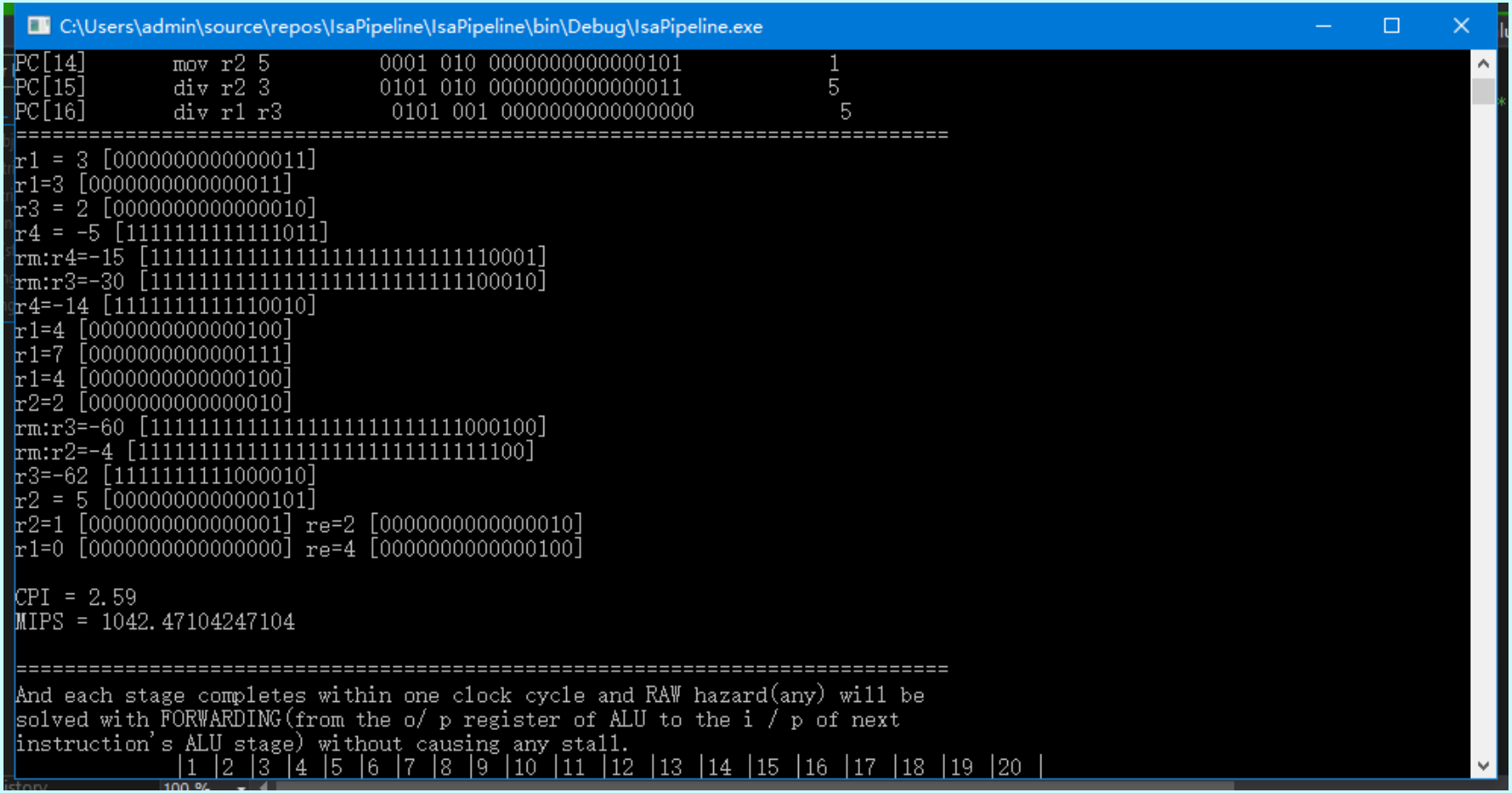
Pipeline core

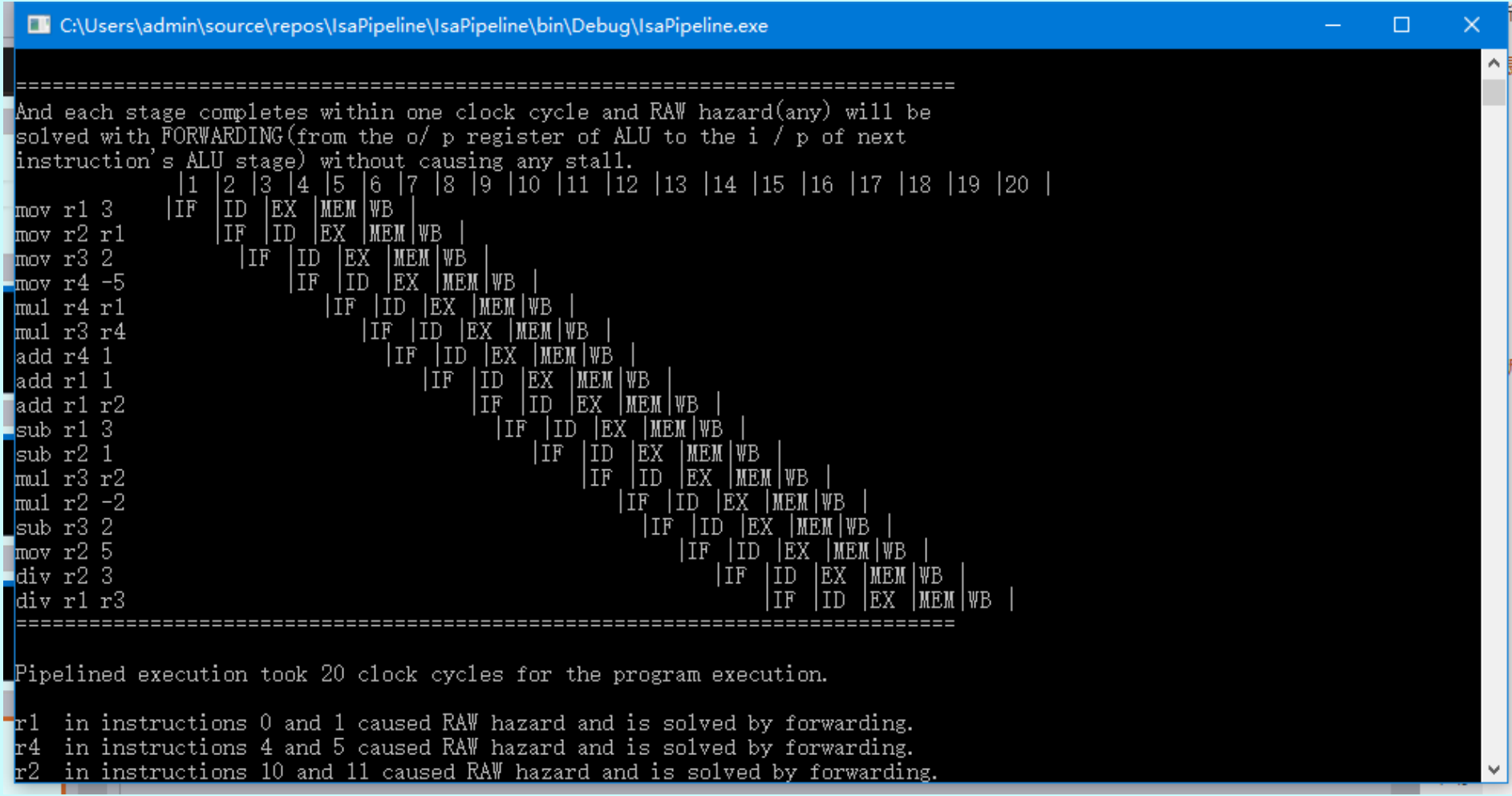
```
string strst     = "              ";
string str_space = "    ";
string estr =      "";
string snum =      "";
int isk =0;
for (int i = 1; i < svc0.GetLength(0)+1 + 3; i++)
{
    snum = snum + i + " |";
    isk = i;
}
Console.Write(strst + "|"  + snum);
string instrk = "";
for (int ik = 0; ik < svc0.GetLength(0)+1; ik++)
{
    for (int jk = 0; jk <= ik - 1; jk++)
    {
        instrk = svc0[jk, 0] + " " + svc0[jk, 1] + " " + svc0[jk, 2];
        estr = str_space + estr;
    }
    Console.Write(instrk + " " + estr + strst);
    Console.WriteLine();
    strst = "|IF |ID |EX |MEM|WB |";
    str_space = "    ";
    estr = "";
}
Console.WriteLine("================================================================================");
```

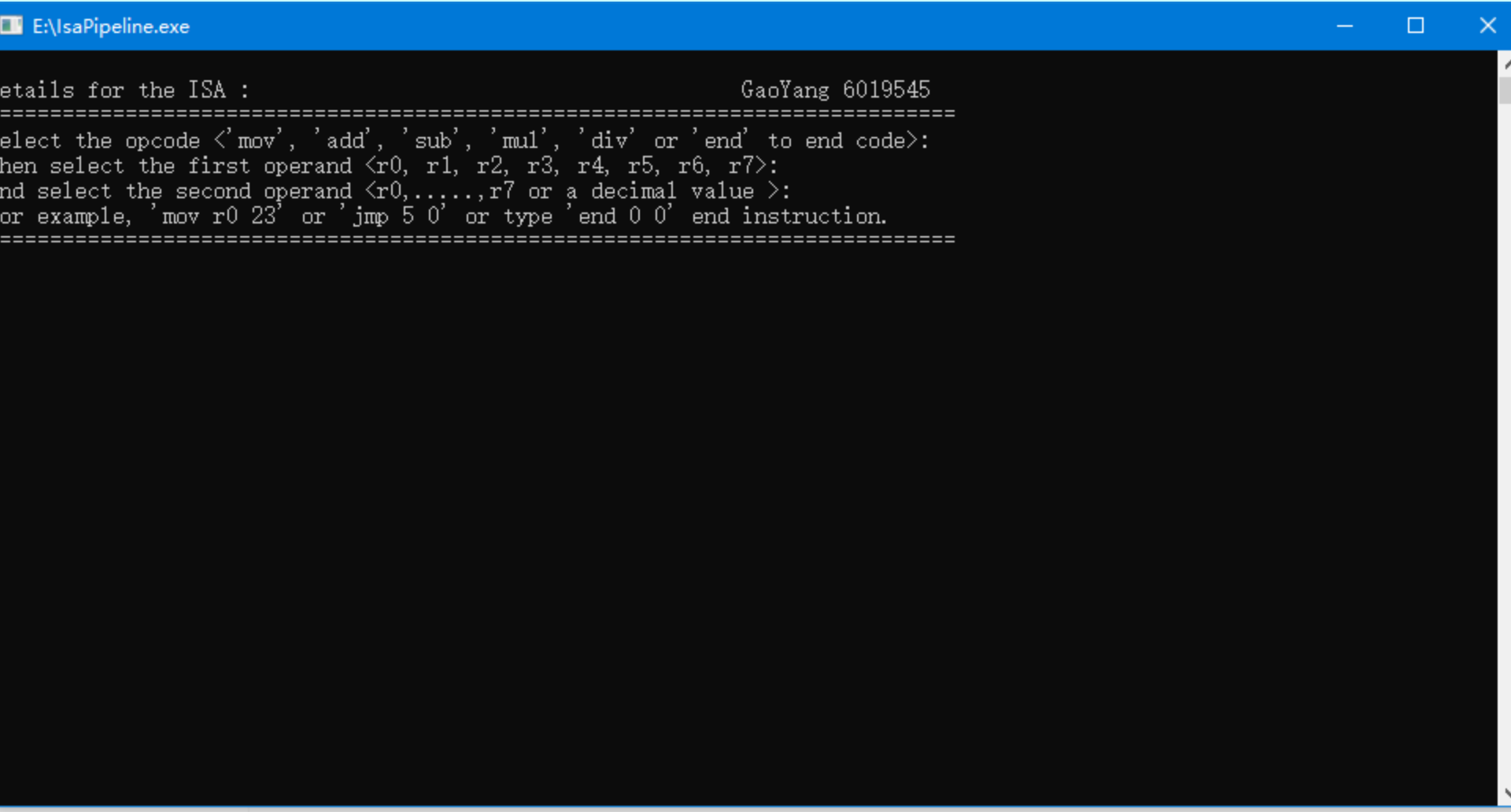
04 Process

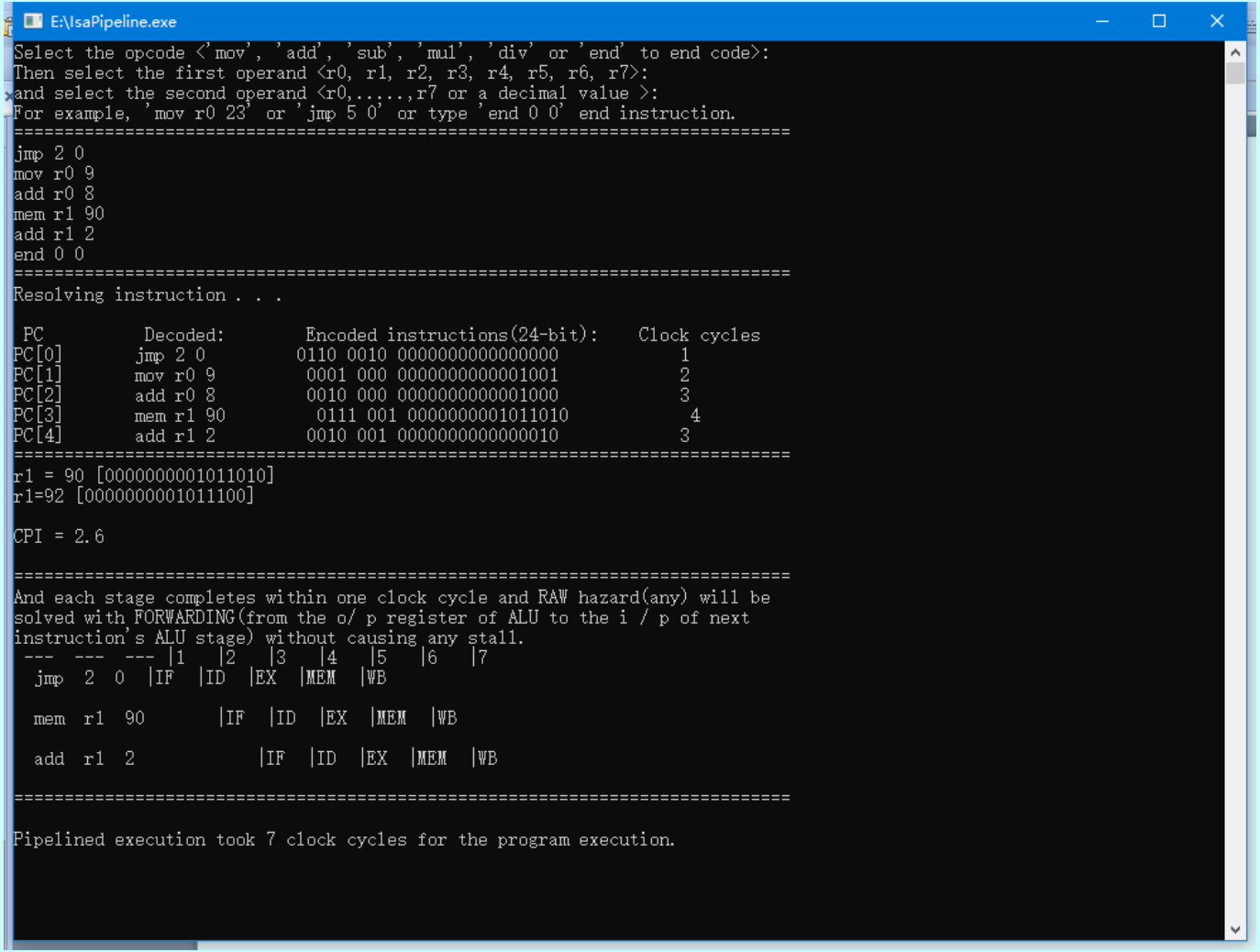

C:\Users\admin\source\repos\IsaPipeline\IsaPipeline\bin\Debug\IsaPipeline.exe
Details for the ISA :
GaoYang 6019545
Select the opcode <'mov', 'add', 'sub', 'mul', 'div' or 'end' to end code>:
Then select the first operand <r0, r1, r2, r3, r4, r5, r6, r7>:
and select the second operand <r0,...,r7 or a decimal value >:
For example, 'mov r0 23' or 'jmp 5 0' or type 'end 0 0' end instruction.
mov r1 3
mov r2 r1
mov r3 2
mov r4 -5
mul r4 r1
mul r3 r4
add r4 1
add r1 1
add r1 r2
sub r1 3
sub r2 1
mul r3 r2
mul r2 -2
sub r3 2
mov r2 5
div r2 3
div r1 r3
end 0 0
Resolving instruction . . .
PC
Decoded:
Encoded instructions(24-bit):
Clock cycles

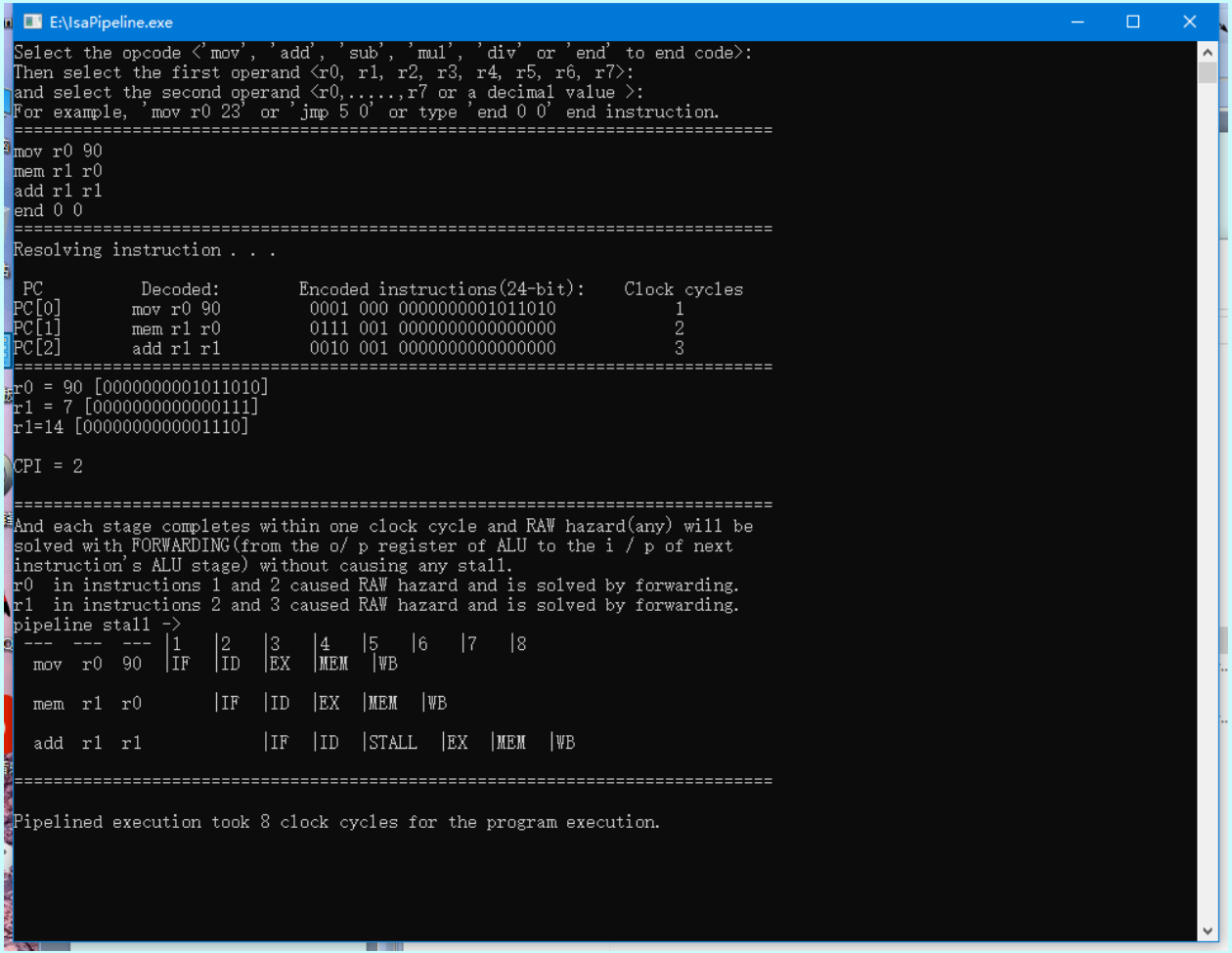

C:\Users\admin\source\repos\IsaPipeline\IsaPipeline\bin\Debug\IsaPipeline.exe
div r1 r3
end 0 0
==========
Resolving instruction . . .
PC Decoded: Encoded instructions(24-bit): Clock cycles
PC[0] mov r1 3 0001 001 000000000000011 1
PC[1] mov r2 r1 0001 010 000000000000000 1
PC[2] mov r3 2 0001 011 000000000000010 1
PC[3] mov r4 -5 0001 100 111111111111011 1
PC[4] mul r4 r1 0100 100 000000000000000 2
PC[5] mul r3 r4 0100 011 000000000000000 2
PC[6] add r4 1 0010 100 000000000000001 3
PC[7] add r1 1 0010 001 000000000000001 3
PC[8] add r1 r2 0010 001 000000000000000 3
PC[9] sub r1 3 0011 001 000000000000011 4
PC[10] sub r2 1 0011 010 000000000000001 4
PC[11] mul r3 r2 0100 011 000000000000000 2
PC[12] mul r2 -2 0100 010 111111111111110 2
PC[13] sub r3 2 0011 011 000000000000010 4
PC[14] mov r2 5 0001 010 000000000000101 1
PC[15] div r2 3 0101 010 000000000000011 5
PC[16] div r1 r3 0101 001 000000000000000 5
==========
r1 = 3 [000000000000011]
r1=3 [000000000000011]
r3 = 2 [000000000000010]
r4 = -5 [111111111111011]
rm:r4=-15 [11111111111111111111111111110001]
rm:r3=-30 [11111111111111111111111111100010]


C:\Users\admin\source\repos\IsaPipeline\IsaPipeline\bin\Debug\IsaPipeline.exe
PC[14] mov r2 5 0001 010 0000000000000101 1
PC[15] div r2 3 0101 010 0000000000000011 5
PC[16] div r1 r3 0101 001 0000000000000000 5
r1 = 3 [0000000000000011]
r1=3 [0000000000000011]
r3 = 2 [0000000000000010]
r4 = -5 [1111111111111011]
rm:r4=-15 [11111111111111111111111111110001]
rm:r3=-30 [11111111111111111111111111100010]
r4=-14 [1111111111110010]
r1=4 [0000000000000100]
r1=7 [0000000000000111]
r1=4 [0000000000000100]
r2=2 [0000000000000010]
rm:r3=-60 [11111111111111111111111111000100]
rm:r2=-4 [11111111111111111111111111111100]
r3=-62 [1111111111000010]
r2 = 5 [0000000000000101]
r2=1 [0000000000000001] re=2 [0000000000000010]
r1=0 [0000000000000000] re=4 [0000000000000100]
CPI = 2.59
MIPS = 1042.47104247104
And each stage completes within one clock cycle and RAW hazard(any) will be
solved with FORWARDING(from the o/ p register of ALU to the i / p of next
instruction's ALU stage) without causing any stall.
|1 |2 |3 |4 |5 |6 |7 |8 |9 |10 |11 |12 |13 |14 |15 |16 |17 |18 |19 |20 |


C:\Users\admin\source\repos\IsaPipeline\IsaPipeline\bin\Debug\IsaPipeline.exe
And each stage completes within one clock cycle and RAW hazard(any) will be
solved with FORWARDING(from the o/ p register of ALU to the i / p of next
instruction's ALU stage) without causing any stall.
|1 |2 |3 |4 |5 |6 |7 |8 |9 |10 |11 |12 |13 |14 |15 |16 |17 |18 |19 |20 |
mov r1 3 |IF |ID |EX |MEM|WB |
mov r2 r1 |IF |ID |EX |MEM|WB |
mov r3 2 |IF |ID |EX |MEM|WB |
mov r4 -5 |IF |ID |EX |MEM|WB |
mul r4 r1 |IF |ID |EX |MEM|WB |
mul r3 r4 |IF |ID |EX |MEM|WB |
add r4 1 |IF |ID |EX |MEM|WB |
add r1 1 |IF |ID |EX |MEM|WB |
add r1 r2 |IF |ID |EX |MEM|WB |
sub r1 3 |IF |ID |EX |MEM|WB |
sub r2 1 |IF |ID |EX |MEM|WB |
mul r3 r2 |IF |ID |EX |MEM|WB |
mul r2 -2 |IF |ID |EX |MEM|WB |
sub r3 2 |IF |ID |EX |MEM|WB |
mov r2 5 |IF |ID |EX |MEM|WB |
div r2 3 |IF |ID |EX |MEM|WB |
div r1 r3 |IF |ID |EX |MEM|WB |
Pipelined execution took 20 clock cycles for the program execution.
r1 in instructions 0 and 1 caused RAW hazard and is solved by forwarding.
r4 in instructions 4 and 5 caused RAW hazard and is solved by forwarding.
r2 in instructions 10 and 11 caused RAW hazard and is solved by forwarding.


E:\IsaPipeline.exe
etails for the ISA :
GaoYang 6019545
elect the opcode <'mov', 'add', 'sub', 'mul', 'div' or 'end' to end code>:
hen select the first operand <r0, r1, r2, r3, r4, r5, r6, r7>:
nd select the second operand <r0,.....,r7 or a decimal value >:
or example, 'mov r0 23' or 'jmp 5 0' or type 'end 0 0' end instruction.


E:\IsaPipeline.exe
Select the opcode <'mov', 'add', 'sub', 'mul', 'div' or 'end' to end code>:
Then select the first operand <r0, r1, r2, r3, r4, r5, r6, r7>:
and select the second operand <r0,.....,r7 or a decimal value >:
For example, 'mov r0 23' or 'jmp 5 0' or type 'end 0 0' end instruction.
jmp 2 0
mov r0 9
add r0 8
mem r1 90
add r1 2
end 0 0
Resolving instruction . . .
PC Decoded: Encoded instructions(24-bit): Clock cycles
PC[0] jmp 2 0 0110 0010 0000000000000000 1
PC[1] mov r0 9 0001 000 0000000000001001 2
PC[2] add r0 8 0010 000 0000000000001000 3
PC[3] mem r1 90 0111 001 0000000001011010 4
PC[4] add r1 2 0010 001 0000000000000010 3
r1 = 90 [0000000001011010]
r1=92 [0000000001011100]
CPI = 2.6
And each stage completes within one clock cycle and RAW hazard(any) will be
solved with FORWARDING(from the o/ p register of ALU to the i / p of next
instruction's ALU stage) without causing any stall.
--- --- --- |1 |2 |3 |4 |5 |6 |7
jmp 2 0 |IF |ID |EX |MEM |WB
mem r1 90 |IF |ID |EX |MEM |WB
add r1 2 |IF |ID |EX |MEM |WB
Pipelined execution took 7 clock cycles for the program execution.


E:\IsaPipeline.exe
Select the opcode <'mov', 'add', 'sub', 'mul', 'div' or 'end' to end code>:
Then select the first operand <r0, r1, r2, r3, r4, r5, r6, r7>:
and select the second operand <r0,.....,r7 or a decimal value >:
For example, 'mov r0 23' or 'jmp 5 0' or type 'end 0 0' end instruction.
mov r0 90
mem r1 r0
add r1 r1
end 0 0
Resolving instruction . . .
PC Decoded: Encoded instructions(24-bit): Clock cycles
PC[0] mov r0 90 0001 000 0000000001011010 1
PC[1] mem r1 r0 0111 001 0000000000000000 2
PC[2] add r1 r1 0010 001 0000000000000000 3
r0 = 90 [0000000001011010]
r1 = 7 [0000000000000111]
r1=14 [0000000000001110]
CPI = 2
And each stage completes within one clock cycle and RAW hazard(any) will be
solved with FORWARDING(from the o/ p register of ALU to the i / p of next
instruction's ALU stage) without causing any stall.
r0 in instructions 1 and 2 caused RAW hazard and is solved by forwarding.
r1 in instructions 2 and 3 caused RAW hazard and is solved by forwarding.
pipeline stall ->
--- --- --- |1 |2 |3 |4 |5 |6 |7 |8
mov r0 90 |IF |ID |EX |MEM |WB
mem r1 r0 |IF |ID |EX |MEM |WB
add r1 r1 |IF |ID |STALL |EX |MEM |WB
Pipelined execution took 8 clock cycles for the program execution.

THANK YOU

GaoYang 6019545