ISA SIMULATION PROGRAM

COMPUTER ARCHITECTURE
CS2202

SIWAPORN CHANRATASSAWAKUL 5913262

DESCRIPTION

- Use Java language
- 24-bit ISA
- Has 8 register > r0 – r7 (16 bits)
- 5 opcodes > mov, add, sub, mul, div

Opcode	Operand 1	Binary number
5 bits	3 bits	16 bits

BINARY AND CLOCK CYCLE OF OPCODE

```
if (opcode.equalsIgnoreCase("mov")) {
    clock = 1;
    noOfOperator = "00001";
```

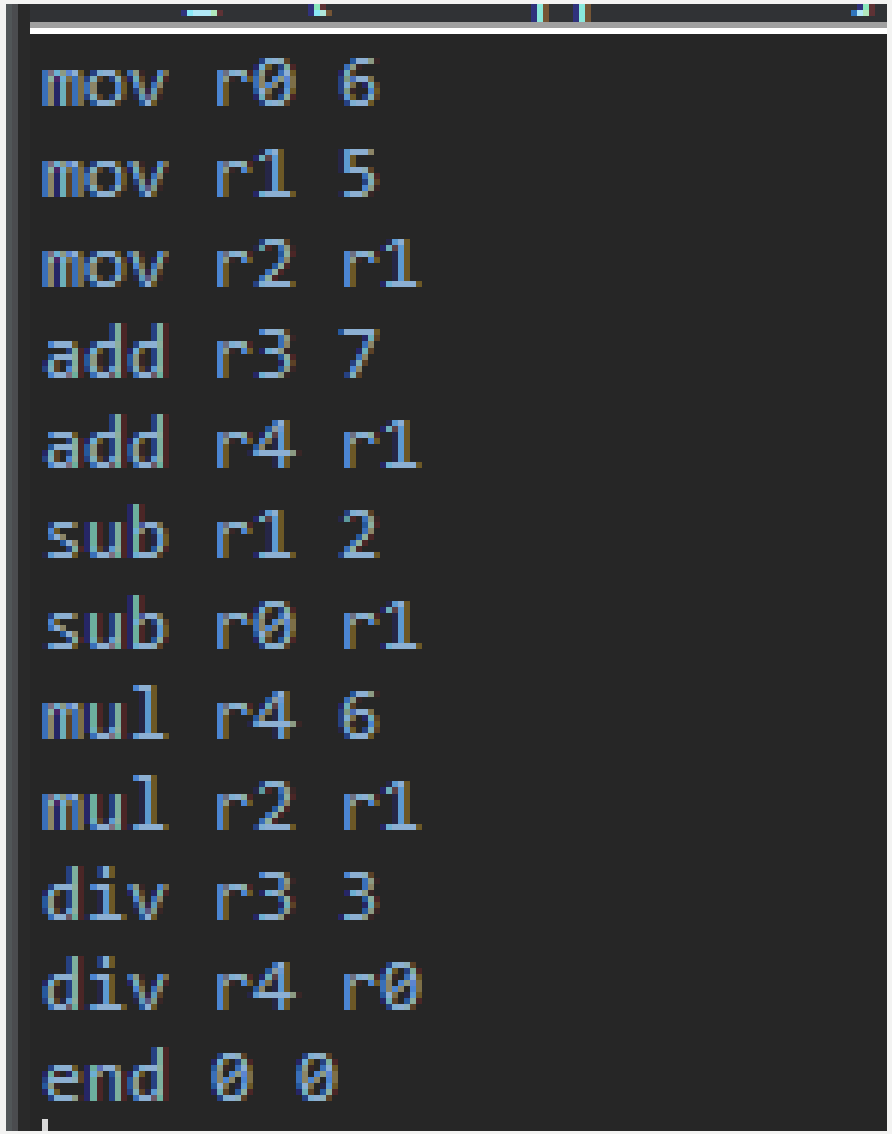
```
else if (opcode.equalsIgnoreCase("add"))
  clock = 2;
  noOfOperator = "00010";
```

```
else if (opcode.equalsIgnoreCase("sub"))
  clock = 2;
  noOfOperator = "00011";
```

```
} else if (opcode.equalsIgnoreCase("mul")) {
    clock = 4;
    noOfOperator = "00100";
```

```
} else if (opcode.equalsIgnoreCase("div")) {
    clock = 4;
    noOfOperator = "00101";
```

INPUT


mov r0 6
mov r1 5
mov r2 r1
add r3 7
add r4 r1
sub r1 2
sub r0 r1
mul r4 6
mul r2 r1
div r3 3
div r4 r0
end 0 0

VALUE OF REGISTER

```
PC[0]   mov r0 , 6   00001 000    0000000000000110
PC[1]   mov r1 , 5   00001 001    0000000000000101
PC[2]   mov r2 , r1   00001 010    0000000000000101
PC[3]   add r3 , 7   00010 011    0000000000000111
PC[4]   add r4 , r1   00010 100    0000000000000101
PC[5]   sub r1 , 2   00011 001    0000000000000010
PC[6]   sub r0 , r1   00011 000    0000000000000011
PC[7]   mul r4 , 6   00100 100    0000000000000110
PC[8]   mul r2 , r1   00100 010    0000000000000011
PC[9]   div r3 , 3   00101 011    0000000000000011
PC[10]   div r4 , r0   00101 100    0000000000000011
```

STEP OF REGISTER

```
r0 = 0000000000000110
r1 = 0000000000000101
r2 = 0000000000000101
r3 = 0000000000000111
r4 = 0000000000000101
r1 = 0000000000000011
r0 = 0000000000000011
rm:r4 = 00000000000000000000000000011110
rm:r2 = 00000000000000000000000000001111
r3 = 0000000000000010 re = 0000000000000001
r4 = 0000000000001010 re = 0000000000000000
```

RESULTS

```
r0   3 [0000000000000011]
r1   3 [0000000000000011]
r2  15 [0000000000001111]
r3   2 [0000000000000010]
r4  10 [0000000000001010]
r5   0 [0000000000000000]
r6   0 [0000000000000000]
r7   0 [0000000000000000]

CPI = 2.4545454545454546
```

SAMPLE CODE

```
72              int integer = Integer.parseInt(registerArr[nOperand], 2);
73
74              if (opcode.equalsIgnoreCase("mov")) {
75                  clock = 1;
76                  noOfOperator = "00001";
77                  if (!isRegist(operand2)) {
78                      registoInt = Integer.parseInt(operand2);
79                      int value = Integer.parseInt(operand2);
80                      toBinary = Integer.toBinaryString(0x10000 | value).substring(1);
81                      if (toBinary.length() > 16) {
82                          toBinary = toBinary.substring(15, toBinary.length());
83                      }
84                      registerArr[nOperand] = toBinary;
85                      op2 = Integer.toBinaryString(0x10000 | registoInt).substring(1);
86                  } else {
87                      String value = registerArr[n];
88                      registerArr[nOperand] = value;
89                      op2 = value;
90                  }
91
92              } else if (opcode.equalsIgnoreCase("add")) {
93                  clock = 2;
94                  noOfOperator = "00010";
95
96                  if (!isRegist(operand2)) {
97                      registoInt = Integer.parseInt(operand2);
98                      result = integer + registoInt;
99                      op2 = Integer.toBinaryString(0x10000 | registoInt).substring(1);
100                 } else {
101                     integer2 = Integer.parseInt(registerArr[n], 2);
102                     result = integer + integer2;
103                     op2 = Integer.toBinaryString(0x10000 | integer2).substring(1);
```

This is what I've already fixed about 2's complement. I used this set of input for testing:

mov r1 8 r1 = 8

mov r2 6 r2 = 6

sub r2 7 r2 = 6 - 7 = -1

mov r3 r2 r3 = r2 = -1

mul r3 2 r3 = -2

mov r4 4 r4 = 4

div r4 2 r4 = 2

end 0 0

The results are showed as in the pictures.

```
mov r1 8
mov r2 6
sub r2 7
mov r3 r2
mul r3 2
mov r4 4
div r4 2
end 0 0
PC[0]  mov r1 , 8  00001 001   0000000000001000   1
PC[1]  mov r2 , 6  00001 010   0000000000000110   1
PC[2]  sub r2 , 7  00011 010   0000000000000111   2
PC[3]  mov r3 , r2  00001 011   1111111111111111   1
PC[4]  mul r3 , 2  00100 011   0000000000000010   4
PC[5]  mov r4 , 4  00001 100   0000000000000100   1
PC[6]  div r4 , 2  00101 100   0000000000000010   4

r1 = 0000000000001000
r2 = 0000000000000110
r2 = 1111111111111111
r3 = 1111111111111111
rm:r3 = 00000000000000001111111111111110
r4 = 0000000000000100
r4 = 0000000000000010 re = 0000000000000000

r0   [0000000000000000]
r1   [0000000000001000]
r2   [1111111111111111]
r3   [1111111111111110]
r4   [0000000000000010]
r5   [0000000000000000]
r6   [0000000000000000]
r7   [0000000000000000]

CPI = 2.0
```