

REPORT LAB04

Rewriting in RISC-V Assembly Language

PB19010450 和泳毅

Design

Use **Jupiter**, an open source and education-oriented RISC-V assembler and runtime simulator.

1. We use three counters to update the number of rocks in rows A, B and C at all times. And they are used to check the status of the game in progress after each time a player has chosen a valid move.
2. In order to check whether the row player input is Row A, we use the **li** instruction to store the ASCII code of the character "A". The same to Row B and Row C.
3. We use a fixed register **t6** to control the order of players. 0 means player 1 and 1 means player 2. Since player 1 starts the game first, we clear **t6** first. After each player ends a valid move, the player is switched by taking the **not** of **t6**.
4. We can use the string "\n" to replace the a newline character (ASCII code x000A) .
5. We use the **ecall** instruction to get the numbers and characters and print the strings. If the **ecall** code is 4, it can print a string whose first address is store in register **a1**. If the **ecall** code is 5, it can get a integer which will be store in register **a0**. If the **ecall** code is 12, it can get a character which will be store in register **a0**.
6. We use the first address of the string which stores the rocks and offsets to control the rocks we want to print at each row.
7. In each loop, we first check the game state to determine if the game is over. Then we check whether the player's input is valid or not. If it is invalid, a hint is performed. If it is valid, make a judgment of the row. Then we perform the calculation operation of the corresponding row. The computation operation consists of updating the number of rocks in each row and adjusting the number of rocks to be printed.

Code Writing

1. Instructions to be used

- **add rd, rs1, rs2**

$x[rd] = x[rs1] + x[rs2]$

- **mv rd rs**

i.e. **addi rd rs 0**, $x[rd] = x[rs1] + 0$

- **andi rd, rs, immediate**

$x[rd] = x[rs] \& \text{sext}(\text{immediate})$

- **not rd,rs**

$x[rd] = \text{not}(x[rs])$

- **sub rd, rs1, rs2**

$x[rd] = x[rs1] - x[rs2]$

- **beq rs1, rs2, LABEL**

Branch, if ($x[rs1] == x[rs2]$) $pc = LABEL$

- **bne rs1, rs2, LABEL**

Branch, if ($x[rs1] \neq x[rs2]$) $pc = LABEL$

- **li rd, immediate**

$x[rd] = \text{immediate}$

- **la rd, LABEL**

$x[rd] = \&LABEL$

- **j LABEL**

$pc = LABEL$

- **jal rd, LABEL**

$x[rd] = pc+4, pc = LABEL$

- **ret**

$pc = x[1]$

2. Initialization

```
1  INIT:
2      andi    t6,t6,0           #Player 1 is 0,Player 2 is 1.
3      andi    x5,x5,0
4      andi    x6,x6,0
5      andi    x7,x7,0
6      li      x2,3              #Rocks in A
7      li      x3,5              #Rocks in B
8      li      x4,8              #Rocks in C
9      li      t3,'A'
10     li      t4,'B'
11     li      t5,'C'
12     li      a0,4
13     la      a1,TITLE
14     ecall
```

3. Check and print the game state

```
1  LOOP:
2      bne     x2,x0,PRINT
3      bne     x3,x0,PRINT
4      beq     x4,x0,CHECK_3     #Game over and someone wins.
5      #
6  PRINT:
7      li      a0,4
8      la      a1,SHOW_A
9      ecall
10     la      a1,SHOW_Ao
11     add     a1,a1,x5
12     ecall
13     li      a0,4
14     la      a1,SHOW_B
15     ecall
16     la      a1,SHOW_Bo
17     add     a1,a1,x6
18     ecall
19     li      a0,4
```

```

20      la      a1,SHOW_C
21      ecall
22      la      a1,SHOW_Co
23      add     a1,a1,x7
24      ecall
25 CHECK_3:
26      bne     t6,x0,P2WINS
27 P1WINS:
28      li      a0,4
29      la      a1,WIN_1
30      ecall
31      j       END
32 P2WINS:
33      li      a0,4
34      la      a1,WIN_2
35      ecall
36 END:
37      li      a0,4
38      la      a1,OVER
39      ecall
40      li      a0, 10          #End the program with status code 0
41      ecall

```

4. Check the player.

```

1 CHECK_1:          #Check the player.
2      bne     t6,x0,PL2
3 PL1:
4      li      a0,4
5      la      a1,SHOW_1
6      ecall
7      j       INPUT
8 PL2:
9      li      a0,4
10     la      a1,SHOW_2
11     ecall

```

5. Check the Row

```

1 CHECK_2:          #Check the rows.
2      beq     x8,t3,INPUTA
3      beq     x8,t4,INPUTB
4      beq     x8,t5,INPUTC
5      j       INV          #Else invalid input
6 #
7 INPUTA:
8      mv      s2,x2
9      mv      s3,x5
10     jal     CAL
11     mv      x2,s2
12     mv      x5,s3
13     not     t6,t6          #Change the player.
14     j       LOOP
15 #
16 INPUTB:
17     mv      s2,x3
18     mv      s3,x6
19     jal     CAL
20     mv      x3,s2

```

```

21         mv        x6,s3
22         not       t6,t6           #Change the player.
23         j         LOOP
24     #
25 INPUTC:
26         mv        s2,x4
27         mv        s3,x7
28         jal       CAL
29         mv        x4,s2
30         mv        x7,s3
31         not       t6,t6           #Change the player.
32         j         LOOP
33     #
34 INV:
35         li        a0,4
36         la        a1,INVALID
37         ecall
38         j         CHECK_1

```

6. Calculate

```

1  CAL:
2      blt        x9,x0,INV
3      beq        x9,x0,INV         #If x9<=0,invalid
4      sub        s2,s2,x9
5      blt        s2,x0,INV
6      add        s3,s3,x9
7      ret

```

7. Non-code section

```

1  TITLE:  .string "\n-----The Game of Nim-----\n"
2  SHOW_A: .string "\n\nROW A: "
3  SHOW_Ao: .string "ooo"
4  SHOW_B: .string "\nROW B: "
5  SHOW_Bo: .string "ooooo"
6  SHOW_C: .string "\nROW C: "
7  SHOW_Co: .string "ooooooooo"
8  SHOW_1: .string "\nPlayer 1, choose a row and number of rocks:"
9  SHOW_2: .string "\nPlayer 2, choose a row and number of rocks:"
10 WIN_1:  .string "\n\nPlayer 1 Wins.\n"
11 WIN_2:  .string "\n\nPlayer 2 Wins.\n"
12 INVALID: .string "\nInvalid move. Try again."
13 OVER:   .string "\n-----Game Over-----\n"

```

Result Test

1.

```

-----The Game of Nim-----

ROW A: ooo
ROW B: ooooo
ROW C: oooooooo
Player 1, choose a row and number of rocks:B
2

ROW A: ooo
ROW B: ooo
ROW C: oooooooo
Player 2, choose a row and number of rocks:A
1

ROW A: oo
ROW B: ooo
ROW C: oooooooo
Player 1, choose a row and number of rocks:C
6

ROW A: oo
ROW B: ooo
ROW C: oo
Player 2, choose a row and number of rocks:A
3

Invalid move. Try again.
Player 2, choose a row and number of rocks:B
3

ROW A: oo
ROW B:
ROW C: oo
Player 1, choose a row and number of rocks:C
6

Invalid move. Try again.
Player 1, choose a row and number of rocks:D
6

Invalid move. Try again.
Player 1, choose a row and number of rocks:C
2

ROW A: oo
ROW B:
ROW C:
Player 2, choose a row and number of rocks:A
1

ROW A: o
ROW B:
ROW C:
Player 1, choose a row and number of rocks:A
1

Player 2 Wins.

-----Game Over-----

```

```

-----The Game of Nim-----

ROW A: ooo
ROW B: ooooo
ROW C: oooooooo
Player 1, choose a row and number of rocks:A
3

ROW A:
ROW B: ooooo
ROW C: oooooooo
Player 2, choose a row and number of rocks:B
5

ROW A:
ROW B:
ROW C: oooooooo
Player 1, choose a row and number of rocks:C
4

ROW A:
ROW B:
ROW C: oooo
Player 2, choose a row and number of rocks:C
2

ROW A:
ROW B:
ROW C: oo
Player 1, choose a row and number of rocks:C
8

Invalid move. Try again.
Player 1, choose a row and number of rocks:C
1

ROW A:
ROW B:
ROW C: o
Player 2, choose a row and number of rocks:A
1

Invalid move. Try again.
Player 2, choose a row and number of rocks:C
1

Player 1 Wins.

-----Game Over-----

```

Appendix

Complete code:

RISC-V:

```

1  .globl main
2
3  .rodata
4  TITLE: .string "\n-----The Game of Nim-----\n"
5  SHOW_A: .string "\n\nROW A: "

```

```

6     SHOW_Ao: .string "ooo"
7     SHOW_B:  .string "\nROW B: "
8     SHOW_Bo: .string "ooooo"
9     SHOW_C:  .string "\nROW C: "
10    SHOW_Co: .string "oooooooo"
11    SHOW_1:  .string "\nPlayer 1, choose a row and number of rocks:"
12    SHOW_2:  .string "\nPlayer 2, choose a row and number of rocks:"
13    WIN_1:   .string "\n\nPlayer 1 Wins.\n"
14    WIN_2:   .string "\n\nPlayer 2 Wins.\n"
15    INVALID: .string "\nInvalid move. Try again."
16    OVER:    .string "\n-----Game Over-----\n"
17
18    .text
19    main:
20    INIT:
21        andi    t6,t6,0          #Player 1 is 0,Player 2 is 1.
22        andi    x5,x5,0
23        andi    x6,x6,0
24        andi    x7,x7,0
25        li      x2,3             #Rocks in A
26        li      x3,5             #Rocks in B
27        li      x4,8             #Rocks in C
28        li      t3,'A'
29        li      t4,'B'
30        li      t5,'C'
31        li      a0,4
32        la      a1,TITLE
33        ecall
34    #
35    LOOP:
36        bne     x2,x0,PRINT
37        bne     x3,x0,PRINT
38        beq     x4,x0,CHECK_3     #Game over and someone wins.
39    #
40    PRINT:
41        li      a0,4
42        la      a1,SHOW_A
43        ecall
44        la      a1,SHOW_Ao
45        add     a1,a1,x5
46        ecall
47        li      a0,4
48        la      a1,SHOW_B
49        ecall
50        la      a1,SHOW_Bo
51        add     a1,a1,x6
52        ecall
53        li      a0,4
54        la      a1,SHOW_C
55        ecall
56        la      a1,SHOW_Co
57        add     a1,a1,x7
58        ecall
59    #
60    CHECK_1:                #Check the player.
61        bne     t6,x0,PL2
62    PL1:
63        li      a0,4
64        la      a1,SHOW_1
65        ecall
66        j       INPUT

```

```

67 PL2:
68     li    a0,4
69     la    a1,SHOW_2
70     ecall
71 #
72 INPUT:
73     li    a0,12
74     ecall
75     mv    x8,a0           #row input
76     li    a0,5
77     ecall
78     mv    x9,a0           #rocks input
79 CHECK_2:                   #Check the rows.
80     beq    x8,t3,INPUTA
81     beq    x8,t4,INPUTB
82     beq    x8,t5,INPUTC
83     j      INV            #Else invalid input
84 #
85 INPUTA:
86     mv    s2,x2
87     mv    s3,x5
88     jal    CAL
89     mv    x2,s2
90     mv    x5,s3
91     not    t6,t6           #Change the player.
92     j      LOOP
93 #
94 INPUTB:
95     mv    s2,x3
96     mv    s3,x6
97     jal    CAL
98     mv    x3,s2
99     mv    x6,s3
100    not    t6,t6           #Change the player.
101    j      LOOP
102 #
103 INPUTC:
104     mv    s2,x4
105     mv    s3,x7
106     jal    CAL
107     mv    x4,s2
108     mv    x7,s3
109     not    t6,t6           #Change the player.
110     j      LOOP
111 #
112 INV:
113     li    a0,4
114     la    a1,INVALID
115     ecall
116     j      CHECK_1
117 #
118 CHECK_3:
119     bne    t6,x0,P2WINS
120 P1WINS:
121     li    a0,4
122     la    a1,WIN_1
123     ecall
124     j      END
125 P2WINS:
126     li    a0,4
127     la    a1,WIN_2

```



```
128         ecall
129 END:
130         li      a0,4
131         la      a1,OVER
132         ecall
133         li      a0, 10          #Ends the program with status code 0
134         ecall
135 #
136 CAL:
137         blt     x9,x0,INV
138         beq     x9,x0,INV      #If x9<=0,invalid
139         sub     s2,s2,x9
140         blt     s2,x0,INV
141         add     s3,s3,x9
142         ret
```
