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] & sext(immediate)
   - **not rd,rs**

     x[rd] = not (x[rs])
   - **sub rd, rs1, rs2**

x[rd] = x[rs1] − x[rs2]

- **beq rs1, rs2, LABLE**

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

- **bne rs1, rs2, LABLE**

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

- **li rd, immediate**

  x[rd] = 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

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

3. Check and print the game state

```
LOOP:
        bne     x2,x0,PRINT
        bne     x3,x0,PRINT
        beq     x4,x0,CHECK_3     #Game over and someone wins.
#
PRINT:
        li      a0,4
        la      a1,SHOW_A
        ecall
        la      a1,SHOW_Ao
        add     a1,a1,x5
        ecall
        li      a0,4
        la      a1,SHOW_B
        ecall
        la      a1,SHOW_Bo
        add     a1,a1,x6
        ecall
        li      a0,4
```

```
        la      a1,SHOW_C
        ecall
        la      a1,SHOW_Co
        add     a1,a1,x7
        ecall
CHECK_3:
        bne     t6,x0,P2WINS
P1WINS:
        li      a0,4
        la      a1,WIN_1
        ecall
        j       END
P2WINS:
        li      a0,4
        la      a1,WIN_2
        ecall
END:
        li      a0,4
        la      a1,OVER
        ecall
        li      a0, 10              #End the program with status code 0
        ecall
```

4. Check the player.

```
CHECK_1:                            #Check the player.
        bne     t6,x0,PL2
PL1:
        li      a0,4
        la      a1,SHOW_1
        ecall
        j       INPUT
PL2:
        li      a0,4
        la      a1,SHOW_2
        ecall
```

5. Check the Row

```
CHECK_2:                            #Check the rows.
        beq     x8,t3,INPUTA
        beq     x8,t4,INPUTB
        beq     x8,t5,INPUTC
        j       INV                 #Else invalid input
#
INPUTA:
        mv      s2,x2
        mv      s3,x5
        jal     CAL
        mv      x2,s2
        mv      x5,s3
        not     t6,t6               #Change the player.
        j       LOOP
#
INPUTB:
        mv      s2,x3
        mv      s3,x6
        jal     CAL
        mv      x3,s2
```

```
        mv      x6,s3
        not     t6,t6           #Change the player.
        j       LOOP
#
INPUTC:
        mv      s2,x4
        mv      s3,x7
        jal     CAL
        mv      x4,s2
        mv      x7,s3
        not     t6,t6           #Change the player.
        j       LOOP
#
INV:
        li      a0,4
        la      a1,INVALID
        ecall
        j       CHECK_1
```

6. Calculate

```
CAL:
        blt     x9,x0,INV
        beq     x9,x0,INV       #If x9<=0,invalid
        sub     s2,s2,x9
        blt     s2,x0,INV
        add     s3,s3,x9
        ret
```

7. Non-code section

```
  TITLE:   .string "\n-------The Game of Nim-------\n"
  SHOW_A:  .string "\n\nROW A: "
  SHOW_Ao: .string "ooo"
  SHOW_B:  .string "\nROW B: "
  SHOW_Bo: .string "ooooo"
  SHOW_C:  .string "\nROW C: "
  SHOW_Co: .string "oooooooo"
  SHOW_1:  .string "\nPlayer 1, choose a row and number of rocks:"
  SHOW_2:  .string "\nPlayer 2, choose a row and number of rocks:"
  WIN_1:   .string "\n\nPlayer 1 Wins.\n"
  WIN_2:   .string "\n\nPlayer 2 Wins.\n"
  INVALID: .string "\nInvalid move. Try again."
  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-------
```

2.

```
-------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：

```
.globl main

.rodata
  TITLE:   .string "\n-------The Game of Nim-------\n"
  SHOW_A:  .string "\n\nROW A: "
```

```
  SHOW_Ao: .string "ooo"
  SHOW_B:  .string "\nROW B: "
  SHOW_Bo: .string "ooooo"
  SHOW_C:  .string "\nROW C: "
  SHOW_Co: .string "oooooooo"
  SHOW_1:  .string "\nPlayer 1, choose a row and number of rocks:"
  SHOW_2:  .string "\nPlayer 2, choose a row and number of rocks:"
  WIN_1:   .string "\n\nPlayer 1 Wins.\n"
  WIN_2:   .string "\n\nPlayer 2 Wins.\n"
  INVALID: .string "\nInvalid move. Try again."
  OVER:    .string "\n-------Game Over-------\n"

.text
main:
INIT:
        andi    t6,t6,0        #Player 1 is 0,Player 2 is 1.
        andi    x5,x5,0
        andi    x6,x6,0
        andi    x7,x7,0
        li      x2,3            #Rocks in A
        li      x3,5            #Rocks in B
        li      x4,8            #Rocks in C
        li      t3,'A'
        li      t4,'B'
        li      t5,'C'
        li      a0,4
        la      a1,TITLE
        ecall
#
LOOP:
        bne     x2,x0,PRINT
        bne     x3,x0,PRINT
        beq     x4,x0,CHECK_3     #Game over and someone wins.
#
PRINT:
        li      a0,4
        la      a1,SHOW_A
        ecall
        la      a1,SHOW_Ao
        add     a1,a1,x5
        ecall
        li      a0,4
        la      a1,SHOW_B
        ecall
        la      a1,SHOW_Bo
        add     a1,a1,x6
        ecall
        li      a0,4
        la      a1,SHOW_C
        ecall
        la      a1,SHOW_Co
        add     a1,a1,x7
        ecall
#
CHECK_1:                          #Check the player.
        bne     t6,x0,PL2
PL1:
        li      a0,4
        la      a1,SHOW_1
        ecall
        j       INPUT
```

```
PL2:
        li      a0,4
        la      a1,SHOW_2
        ecall
#
INPUT:
        li      a0,12
        ecall
        mv      x8,a0           #row input
        li      a0,5
        ecall
        mv      x9,a0           #rocks input
CHECK_2:                        #Check the rows.
        beq     x8,t3,INPUTA
        beq     x8,t4,INPUTB
        beq     x8,t5,INPUTC
        j       INV             #Else invalid input
#
INPUTA:
        mv      s2,x2
        mv      s3,x5
        jal     CAL
        mv      x2,s2
        mv      x5,s3
        not     t6,t6           #Change the player.
        j       LOOP
#
INPUTB:
        mv      s2,x3
        mv      s3,x6
        jal     CAL
        mv      x3,s2
        mv      x6,s3
        not     t6,t6           #Change the player.
        j       LOOP
#
INPUTC:
        mv      s2,x4
        mv      s3,x7
        jal     CAL
        mv      x4,s2
        mv      x7,s3
        not     t6,t6           #Change the player.
        j       LOOP
#
INV:
        li      a0,4
        la      a1,INVALID
        ecall
        j       CHECK_1
#
CHECK_3:
        bne     t6,x0,P2WINS
P1WINS:
        li      a0,4
        la      a1,WIN_1
        ecall
        j       END
P2WINS:
        li      a0,4
        la      a1,WIN_2
```

```
        ecall
END:
        li      a0,4
        la      a1,OVER
        ecall
        li      a0, 10              #Ends the program with status code 0
        ecall
#
CAL:
        blt     x9,x0,INV
        beq     x9,x0,INV           #If x9<=0,invalid
        sub     s2,s2,x9
        blt     s2,x0,INV
        add     s3,s3,x9
        ret
```