Eight Queens on MIPS

> Zhixun TAN
> EE 25
> 2012011120

The basic idea is putting the queens one row after another. First, we know that any two queens can't be in the same row, which indicates that for any row (0 to 7), there can only be exactly one queen.

For each row, there are 8 positions (namely columns) where we can put the queen. So we may use a number between 0 to 7 to denote the position of the queen in each row. There are 8 rows in all, meaning that we need 8 numbers. Each number requires 3 bits, so 8 numbers require 24 bits in all, which can be stored in one single `word`.

How do we check that not any two queens are in the same column? This requires us to track the previous moves before make the new move. In other words, suppose we are about to put a queen in row 3, col 4, we must first make sure that there are no previous queens in col 4. Thus, we use an 8-bit bitset to denote which columns have been 'used'.

For example, before any queen is put, the bitset is `00000000`. We put a queen in row 0, col 1, then the bitset becomes `01000000`. Then we put a queen in row 1, col 3, the bitset becomes `01010000`. Then we put a queen in row 2, col 1, but wait! -- from the bitset we know that col 1 has been occupied, so we can't do that.

The situation is similar when it comes to diagonals. There are 14 left diagonals:

```
 7  8  9 10 11 12 13 14
 6  7  8  9 10 11 12 13
 5  6  7  8  9 10 11 12
 4  5  6  7  8  9 10 11
 3  4  5  6  7  8  9 10
 2  3  4  5  6  7  8  9
 1  2  3  4  5  6  7  8
 0  1  2  3  4  5  6  7
```

There are also 14 right diagonals.

Python version of Eight Queens:

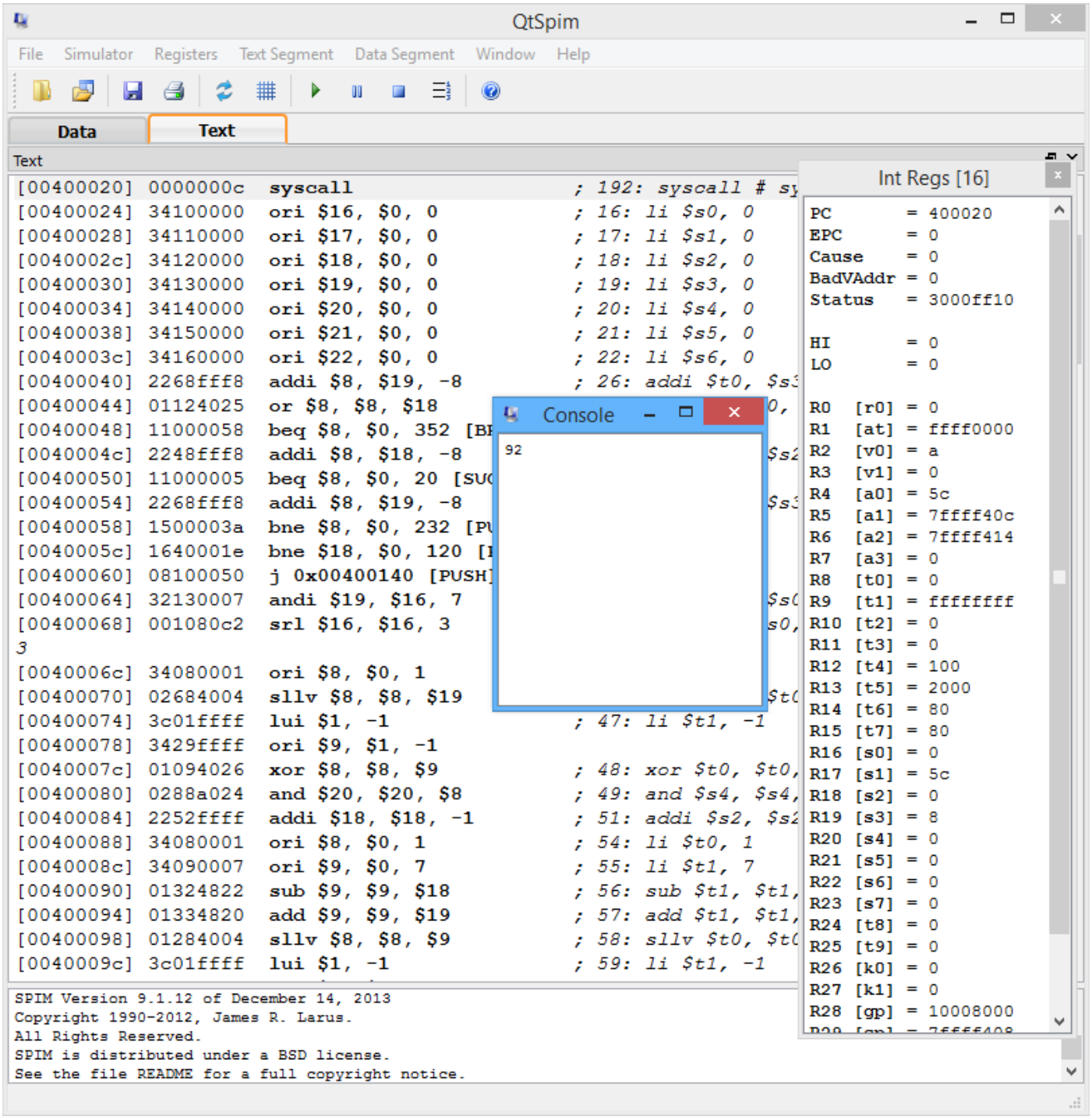
Here we give the python version of the algorithm above.

```
num = 8
curr = 0
```

```
vert = 0
ldiag = 0
rdiag = 0
row = 0
pos = 0
count = 0

while (True):
    if pos == num and row == 0: # BREAK
        break

    if row == num: # success
        print oct(curr)
        pos = curr & 7
        curr = curr >> 3

        row -= 1
        vert = vert & (~(1 << pos))
        ldiag = ldiag & (~(1 << (7 - row + pos)))
        rdiag = rdiag & (~(1 << (row + pos)))
        pos += 1
        count += 1
        continue

    if pos == num and row != 0: # POP
        pos = curr & 7
        curr = curr >> 3

        row -= 1
        vert = vert & (~(1 << pos))
        ldiag = ldiag & (~(1 << (7 - row + pos)))
        rdiag = rdiag & (~(1 << (row + pos)))
        pos += 1
        continue

    # PUSH
    # Check the correctness of this move.
    t_vert = 1 << pos
    t_ldiag = 1 << (7 - row + pos)
    t_rdiag = 1 << (row + pos)

    fail = vert & t_vert
    fail = fail | (ldiag & t_ldiag)
    fail = fail | (rdiag & t_rdiag)

    if fail == 0:
        curr = curr << 3
        curr = curr | pos
```

```
        vert = vert | t_vert
        ldiag = ldiag | t_ldiag
        rdiag = rdiag | t_rdiag
        pos = 0
        row += 1

    else:
        pos += 1

print count
```

RESULT


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
Data
Text
Text
[00400020] 0000000c syscall ; 192: syscall # sy
[00400024] 34100000 ori $16, $0, 0 ; 16: li $s0, 0
[00400028] 34110000 ori $17, $0, 0 ; 17: li $s1, 0
[0040002c] 34120000 ori $18, $0, 0 ; 18: li $s2, 0
[00400030] 34130000 ori $19, $0, 0 ; 19: li $s3, 0
[00400034] 34140000 ori $20, $0, 0 ; 20: li $s4, 0
[00400038] 34150000 ori $21, $0, 0 ; 21: li $s5, 0
[0040003c] 34160000 ori $22, $0, 0 ; 22: li $s6, 0
[00400040] 2268fff8 addi $8, $19, -8 ; 26: addi $t0, $s3
[00400044] 01124025 or $8, $8, $18
[00400048] 11000058 beq $8, $0, 352 [BE
[0040004c] 2248fff8 addi $8, $18, -8
[00400050] 11000005 beq $8, $0, 20 [SUC
[00400054] 2268fff8 addi $8, $19, -8
[00400058] 1500003a bne $8, $0, 232 [PU
[0040005c] 1640001e bne $18, $0, 120 [E
[00400060] 08100050 j 0x00400140 [PUSH]
[00400064] 32130007 andi $19, $16, 7
[00400068] 001080c2 srl $16, $16, 3
3
[0040006c] 34080001 ori $8, $0, 1
[00400070] 02684004 sllv $8, $8, $19
[00400074] 3c01ffff lui $1, -1 ; 47: li $t1, -1
[00400078] 3429ffff ori $9, $1, -1
[0040007c] 01094026 xor $8, $8, $9 ; 48: xor $t0, $t0,
[00400080] 0288a024 and $20, $20, $8 ; 49: and $s4, $s4,
[00400084] 2252ffff addi $18, $18, -1 ; 51: addi $s2, $s2
[00400088] 34080001 ori $8, $0, 1 ; 54: li $t0, 1
[0040008c] 34090007 ori $9, $0, 7 ; 55: li $t1, 7
[00400090] 01324822 sub $9, $9, $18 ; 56: sub $t1, $t1,
[00400094] 01334820 add $9, $9, $19 ; 57: add $t1, $t1,
[00400098] 01284004 sllv $8, $8, $9 ; 58: sllv $t0, $t0
[0040009c] 3c01ffff lui $1, -1 ; 59: li $t1, -1
Console
92
Int Regs [16]
PC = 400020
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = ffff0000
R2 [v0] = a
R3 [v1] = 0
R4 [a0] = 5c
R5 [a1] = 7ffff40c
R6 [a2] = 7ffff414
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = ffffffff
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 100
R13 [t5] = 2000
R14 [t6] = 80
R15 [t7] = 80
R16 [s0] = 0
R17 [s1] = 5c
R18 [s2] = 0
R19 [s3] = 8
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
SPIM Version 9.1.12 of December 14, 2013
Copyright 1990-2012, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.