Solution

Problem 1: (14 points)

[1] 1000 [2] 0000 0100 [3] 0001
[4] 0000 0010 [5] 1110 0001 [6] 0000 0100
[7] 0001 0010

Problem 2: (12 points)

[1] %rax
[2] 0x00000000 00004001
[3] 0x4010
[4] 0x00000000 00000011
[5] %rdx
[6] 0xffffffff aabbccdd
[7] -- or CF,ZF,SF,OF
[8] -- or 0,0,0,0
[9] %rdx
[10] 0x00000000 00004000
[11] %rcx, %rsp
[12] 0x0804ff80 00000000, 0x00000000 00004018

Problem 3: (18 points)

1. [1] 120 [2] 104
[3] 56 [4] 104
[5] 0x601068 [6] 0x601070
[7] 0x601090 [8] 0x601070
[9] 0x60107e [10] 0x6010d0

2. 104 - (8+1+8+14+64) = 9

3. 8 bytes. It will use 96 bytes at least.
(For example, put loaded after keys and leave 1 byte padding after that)

4. 128 bytes.

Problem 4: (11 points)

1 [1] 0x601088
[2] 0x6010b4 + 4n
[3] 0x601068 + 28n

2. a: -0x40(%rbp)
b: -0x60(%rbp)

3. [1] 2 [2] 1
4. Line 16: 6, %rax = row*4+i (i = 2, row = 1)
   Line 23: 6, %rax = i*2+col (i = 2, col = 2)

Problem 5: (22 points)

1 [1] result>8?2:result [2] '4'

[3] '3' [4] result + i

[5] result * 2 - 1 [6] %esi (%rsi is incorrect)

[7] 0 or NONE [8] .L5(,%rax or %eax,8)

[9] -24(%rbp) [10] NONE or jmp .L3 or nop

[11] jl .L13 [12] -8(%rbp)

2 pos:3

0xab89

3 pos:3

0x4567

Problem 6: (23 points)

1 [1] addl $16,%rsp

2 [1] 0x40052d [2] movl %eax,0x18(%rbp)

3 [1] 6 [2] 0x1000df20

[3] 0x1000df20 [4] 4

[5] 6 [6] 0x40057c

[7] 0x1000df40

[8] 5 [9] 8

[10] 2 [11] 7

[12] 1 [13] 3

[14] 6 [15] 4

4 [16] ((long*)&h)[-2] (or ((long *)&g)[-1])