HW2 solution

2.9

2.9
```
sll   $t0, $s1, 2   # $t0 <-- 4*g
add   $t0, $t0, $s7 # $t0 <-- Addr(B[g])
lw    $t0, 0($t0)   # $t0 <-- B[g]
addi  $t0, $t0, 1   # $t0 <-- B[g]+1
sll   $t0, $t0, 2   # $t0 <-- 4*(B[g]+1) = Addr(A[B[g]+1])
lw    $s0, 0($t0)   # f    <-- A[B[g]+1]
```

2.13

2.13

2.13.1 $128 + \times > 2^{31}-1$, $x > 2^{31}-129$ and $128 + x < -2^{31}$, $x < -2^{31} - 128$ (impossible)

2.13.2 $128 - x > 2^{31}-1$, $x < -2^{31}+129$ and $128 - x < -2^{31}$, $x > 2^{31} + 128$ (impossible)

2.13.3 $x - 128 < -2^{31}$, $x < -2^{31} + 128$ and $x - 128 > 2^{31} - 1$, $x > 2^{31} + 127$ (impossible)

2.18

2.18

2.18.1 opcode would be 8 bits, `rs`, `rt`, `rd` fields would be 7 bits each

2.18.2 opcode would be 8 bits, `rs` and `rt` fields would be 7 bits each

2.18.3 more registers → more bits per instruction → could increase code size

more registers → less register spills → less instructions

more instructions → more appropriate instruction → decrease code size

more instructions → larger opcodes → larger code size

2.19

2.19

2.19.1 `0xBABEFEF8`

2.19.2 `0xAAAAAAA0`

2.19.3 `0x00005545`

2.19.1 題目有誤，所以下兩種版本得到的答案皆對

題目	sll $t2, $t0, 44	sll $t2, $t0, 4
ANS	0x12345678	0xBABEFEF8

2.22

2.22
```
lw  $t3, 0($s1)
sll $t1, $t3, 4
```

2.26

2.26

2.26.1 20

2.26.2
```
i = 10;
do {
    B += 2;
    i = i - 1;
} while ( i > 0)
```

2.26.3 5*N

2.26.3 寫 5*N+2 也可以

2.31

2.31
```
fib:   addi $sp, $sp, -12       # make room on stack
       sw   $ra, 8($sp)         # push $ra
       sw   $s0, 4($sp)         # push $s0
       sw   $a0, 0($sp)         # push $a0 (N)
       bgt  $a0, $0, test2      # if n>0, test if n=1
       add  $v0, $0, $0         # else fib(0) = 0
       j rtn                    #
test2: addi $t0, $0, 1          #
       bne  $t0, $a0, gen       # if n>1, gen
       add  $v0, $0, $t0        # else fib(1) = 1
       j rtn
gen:   subi $a0, $a0,1          # n-1
       jal  fib                 # call fib(n-1)
       add  $s0, $v0, $0        # copy fib(n-1)
       sub  $a0, $a0,1          # n-2
       jal  fib                 # call fib(n-2)
       add  $v0, $v0, $s0       # fib(n-1)+fib(n-2)
rtn:   lw   $a0, 0($sp)         # pop $a0
       lw   $s0, 4($sp)         # pop $s0
       lw   $ra, 8($sp)         # pop $ra
       addi $sp, $sp, 12        # restore sp
       jr   $ra

# fib(0) = 12 instructions, fib(1) = 14 instructions,
# fib(N) = 26 + 18N instructions for N >=2
```

2.33 這題請以 2.31 的 code 來討論 stack 中要存哪些東西

```
2.33 after calling function fib:
     old $sp ->  0x7ffffffc    ???
                 -4            contents of register $ra for
                               fib(N)
                 -8            contents of register $s0 for
                               fib(N)
     $sp->       -12           contents of register $a0 for
                               fib(N)
     there will be N-1 copies of $ra, $s0 and $a0
```

2.43 sc 指令成功則$t1 會回傳 1，藉由判斷$t1 的值決定是否再執行一次 lock()

```
2.43 trylk: li   $t1,1
            ll   $t0,0($a0)
            bnez $t0,trylk
            sc   $t1,0($a0)
            beqz $t1,trylk
            lw   $t2,0($a1)
            slt  $t3,$t2,$a2
            bnez $t3,skip
            sw   $a2,0($a1)
     skip:  sw   $0,0($a0)
```