CO 2021-Fall HW2 Solution

2.3

```
sub     $t0, $s3, $s4  # $t0=i-j
sll     $t0, $t0, 2    # $t0=4*(i-j)
add     $t0, $s6, $t0  # $t0=&A[i-j]
lw      $t1, 0($t0)    # $t1=A[i-j]
sw      $t1, 32($s7)   # B[8]=A[i-j]
```

2.4

```
sll     $t0, $s0, 2    # $t0=f*4
add     $t0, $s6, $t0  # $t0=&A[f]
sll     $t1, $s1, 2    # $t1=g*4
add     $t1, $s7, $t1  # $t1=&B[g]
lw      $s0, 0($t0)    # f=A[f]
addi    $t2, $t0, 4    # $t2=&A[f+1]
lw      $t0, 0($t2)    # $t0=A[f+1]
add     $t0, $t0, $s0  # $t0=A[f]+A[f+1]
sw      $t0, 0($t1)    # B[g]=A[f]+A[f+1]
```

→ B[g] = A[f]+A[f+1]

2.7 若考出來以 MIPS-32 為準。

for 64 bits MIPS system(8-byte per word):

```
sll     $t0, $s3, 3    # $t0=i*8
add     $t0, $t0, $s6  # $t0=&A[i]
lw      $t1, 0($t0)    # $t1=A[i]
sll     $t2, $s4, 3    # $t2=j*8
add     $t2, $t2, $s6  # $t2=&A[j]
lw      $t3, 0($t2)    # $t3=A[j]
add     $t1, $t1, $t3  # $t1=A[i]+A[j]
sw      $t1, 64($s7)   # B[8]=A[i]+A[j]
```

for 32 bits MIPS system(4-byte per word):

```
sll     $t0, $s3, 2    # $t0=i*4
add     $t0, $t0, $s6  # $t0=&A[i]
lw      $t1, 0($t0)    # $t1=A[i]
sll     $t2, $s4, 2    # $t2=j*4
add     $t2, $t2, $s6  # $t2=&A[j]
lw      $t3, 0($t2)    # $t3=A[j]
add     $t1, $t1, $t3  # $t1=A[i]+A[j]
sw      $t1, 32($s7)   # B[8]=A[i]+A[j]
```

2.8

```
addi    $t0, $s6, 4    # $t0=&A[1]
add     $t1, $s6, $0   # $t1=&A
sw      $t1, 0($t0)    # A[1]=&A
lw      $t0, 0($t0)    # $t0=A[1]=&A
add     $s0, $t1, $t0  # f=&A+&A=2*(&A)
```

→ f=2*(&A)

2.9

	type	opcode	rs	rt	rd	immed
addi $t0, $s6, 4	I-type	8	22	8		4
add $t1, $s6, $0	R-type	0	22	0	9	
sw $t1, 0($t0)	I-type	43	8	9		0
lw $t0, 0($t0)	I-type	35	8	8		0
add $s0, $t1, $t0	R-type	0	9	8	16	

2.10

2.10.1

$s0 = 0x8000_0000

$s1 = 0xD000_0000

$s0 + $s1 = 0x8000_0000 + 0xD000_0000 = 0x1_5000_0000

→ overflow, $t0 = 0x5000_0000

2.10.2

overflow

2.10.3

$s0 = 0x8000_0000

$s1 = 0xD000_0000

$s0 - $s1 = 0x8000_0000 - 0xD000_0000

= 0x8000_0000 + 0x3000_0000 = 0xB000_0000

→ no overflow, $t0 = 0xB000_0000

2.10.4

desired result

2.10.5

$s0 = 0x8000_0000

$s1 = 0xD000_0000

$s0 + $s1 = 0x8000_0000 + 0xD000_0000 = 0x1_5000_0000

$t0 = 0x5000_0000 (overflow)

$t0 + $s0 = 0x5000_0000 + 0x8000_0000 = 0xD000_0000

→ overflow, $t0 = 0xD000_0000

2.10.6

Overflow

2.17

2.17.1

```
sll $t2, $t0, 4   => 0xAAAA_AAA0
or  $t2, $t2, $t1 => 0xAAAA_AAA0 (bitwise or) 0x1234_5678 = 0xBABE_FEF8
```

2.17.2

```
sll  $t2, $t0,  4  => 0xAAAA_AAA0
andi $t2, $t2, -1 => 0xAAAA_AAA0 (bitwise and) 0xFFFF_FFFF = 0xAAAA_AAA0
```

2.17.3

```
srl  $t2, $t0, 3      => 0x1555_5555
andi $t2, $t2, 0xFFEF => 0x1555_5555 (bitwise and) 0x0000_FFEF = 0x0000_5545
```

2.24

2.24.1

20 (loop “addi $s2, $s2, 2” for 10 times)

2.24.2

相同概念的 C code 都會給分。

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

2.24.3

5N+2 (5N 為完整的 loop，+2 為脫離 loop 時的 slt + beq)

2.25

此為解答給的參考 MIPS code，等效且指令數少於此 code 的都會給分。

```
        addi    $t0,    $0,     0       # i = 0;
        beq     $0,     $0,     TEST1   # goto TEST1;
LOOP1:  addi    $t1,    $0,     0       # j = 0;
        beq     $0,     $0,     TEST2   # goto TEST2;
LOOP2:  add     $t3,    $t0,    $t1     # $t3 = i + j;
        sll     $t2,    $t1,    4       # j = j << 4; for 4*j and 4byte offset
        add     $t2,    $t2,    $s2     # $t2 = D + 4 * j * 4;
        sw      $t3,    0($t2)          # D[4*j] = i + j;
        addi    $t1,    $t1,    1       # j++;
TEST2:  slt     $t2,    $t1,    $s1     # $t2 = j < b
        bne     $t2,    $0,     LOOP2   # if ($t2 != 0) goto LOOP2
        addi    $t0,    $t0,    1       # i++;
TEST1:  slt     $t2,    $t0,    $s0     # $t2 = i < a;
        bne     $t2,    $0,     LOOP1   # if ($t2 != 0) goto LOOP1;
```

2.26

outer loop repeat 10 times, inner loop executed once.

Initial inst (2 insts):

```
    addi    $t0,    $0,     0
    beq     $0,     $0,     TEST1
```

#exe. inst each outer loop:

14 (from TEST1 to TEST1)

Exit loop(2 insts):

```
    slt     $t2,    $t0,    $s0
    bne     $t2,    $0,     LOOP1
```

```
total #inst: 2 + 14 * 10 + 2 = 144
```

14 instructions to implement and 144 instructions executed.

2.29

Stack pointer間隔16或12均可。

```
fib:    addi    $sp,    $sp,    -16     # 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,    16      # restore sp
        jr      $ra
```