Solution 2.6

2.6.1 (b)

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

2.6.2 (b) 9

2.6.3 (b) 6

2.6.4

a.	`f = f - i;`
b.	`f = 2 * (&A);`

2.6.5

a.	$s0 = –30
b.	$s0 = 512

2.6.6

a.

	Type	opcode	rs	rt	rd	immed
`sub $s0, $s0, $s1`	R-type	0	16	17	16	
`sub $s0, $s0, $s3`	R-type	0	16	19	16	
`add $s0, $s0, $s1`	R-type	0	16	17	16	

b.

	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	

Solution 2.8

2.8.1

a.	50000000, overflow
b.	0, no overflow

2.8.2

a.	B0000000, no overflow
b.	2, no overflow

2.8.3

a.	D0000000, overflow
b.	000000001, no overflow

Solution 2.11

2.11.1

a.	$0000\ 0001\ 0000\ 1000\ 0100\ 0000\ 0010\ 0000_{two}$
b.	$0000\ 0010\ 0101\ 0011\ 1000\ 1000\ 0010\ 0010_{two}$

2.11.2

a.	17317920
b.	39028770

2.11.3

a.	`add  $t0, $t0, $t0`
b.	`sub  $s1, $s2, $s3`

Solution 2.20

2.20.1

a.

```
FACT:   addi $sp, $sp, -8  # make room in stack for 2 more items
        sw   $ra, 4($sp)   # save the return address
        sw   $a0, 0($sp)   # save the argument n
        slti $t0, $a0, 1   # $t0 = $a0 x 2
        beq, $t0, $0, L1   # if $t0 = 0, goto L1
        add  $v0, $0, 1    # return 1
        add  $sp, $sp, 8   # pop two items from the stack
        jr   $ra           # return to the instruction after jal
L1:     addi $a0, $a0, -1  # subtract 1 from argument
        jal  FACT          # call fact(n-1)
        lw   $a0, 0($sp)   # just returned from jal: restore n
        lw   $ra, 4($sp)   # restore the return address
        add  $sp, $sp, 8   # pop two items from the stack
        mul  $v0, $a0, $v0 # return n*fact(n-1)
        jr   $ra           # return to the caller
```

2.20.2

a. 13 MIPS instructions to execute non-recursive vs. 15 instructions to execute (corrected version of) recursion

Non-recursive version:

```
FACT:   addi  $sp, $sp, -4
        sw    $ra, 4($sp)
        add   $s0, $0, $a0
        add   $s2, $0, $1
LOOP:   slti  $t0, $s0, 2
        bne   $t0, $0, DONE
        mul   $s2, $s0, $s2
        addi  $s0, $s0, -1
        j LOOP
DONE:   add   $v0, $0, $s2
        lw    $ra, 4($sp)
        addi  $sp, $sp, 4
        jr    $ra
```

2.20.3

a.

```
Recursive version
FACT:   addi  $sp, $sp, -8
        sw    $ra, 4($sp)
        sw    $a0, 0($sp)
        add   $s0, $0, $a0
HERE:   slti  $t0, $a0, 2
        beq   $t0, $0, L1
        addi  $v0, $0, 1
        addi  $sp, $sp, 8
        jr    $ra
L1:     addi  $a0, $a0, -1
        jal   FACT
        mul   $v0, $s0, $v0
        lw    $a0, 0($sp)
        lw    $ra, 4($sp)
        addi  $sp, $sp, 8
        jr    $ra

at label HERE, after calling function FACT with input of 4:
old $sp ->      0xnnnnnnnn      ???
                -4              contents of register $ra
$sp->           -8              contents of register $a0

at label HERE, after calling function FACT with input of 3:
old $sp ->      0xnnnnnnnn      ???
                -4              contents of register $ra
                -8              contents of register $a0
                -12             contents of register $ra
$sp->           -16             contents of register $a0

at label HERE, after calling function FACT with input of 2:
old $sp ->      0xnnnnnnnn      ???
                -4              contents of register $ra
                -8              contents of register $a0
                -12             contents of register $ra
                -16             contents of register $a0
                -20             contents of register $ra
$sp->           -24             contents of register $a0

at label HERE, after calling function FACT with input of 1:
old $sp ->      0xnnnnnnnn      ???
                -4              contents of register $ra
                -8              contents of register $a0
                -12             contents of register $ra
                -16             contents of register $a0
                -20             contents of register $ra
                -24             contents of register $a0
                -28             contents of register $ra
$sp->           -32             contents of register $a0
```

Solution 2.25

2.25.1 Generally, all solutions are similar:

```
lui $t1, top_16_bits
ori $t1, $t1, bottom_16_bits
```

2.25.5 Generally, all solutions are similar:

```
add  $t1, $0, $0             #clear $t1
addi $t2, $0, top_8_bits     #set top 8b
sll  $t2, $t2, 24            #shift left 24 spots
or   $t1, $t1, $t2           #place top 8b into $t1
addi $t2, $0, nxt1_8_bits    #set next 8b
sll  $t2, $t2, 16            #shift left 16 spots
or   $t1, $t1, $t2           #place next 8b into $t1
addi $t2, $0, nxt2_8_bits    #set next 8b
sll  $t2, $t2, 24            #shift left 8 spots
or   $t1, $t1, $t2           #place next 8b into $t1
ori  $t1, $t1, bot_8_bits    #or in bottom 8b
```