CSCI 4593: Homework 2

2.3

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

2.7

Little Endian

Low Memory→High Memory

Hex Digit	12	ef	cd	ab
Address	0	4	8	12

Big Endian

Low Memory→High Memory

Hex Digit	ab	cd	ef	12
Address	0	4	8	12

2.11

instruction	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	

2.12.3

$s0 = 1000 0000 0000 0000 0000 0000 0000 0000

$s1 = 1101 0000 0000 0000 0000 0000 0000 0000

inverted $s1 = 0010 1111 1111 1111 1111 1111 1111 1111

negated $s1 = 0011 0000 0000 0000 0000 0000 0000 0000

$$
\begin{array}{cllllllll}
 & 1000 & 0000 & 0000 & 0000 & 0000 & 0000 & 0000 & 0000 \\
+ & 0011 & 0000 & 0000 & 0000 & 0000 & 0000 & 0000 & 0000 \\
\hline
 & 1011 & 0000 & 0000 & 0000 & 0000 & 0000 & 0000 & 0000
\end{array}
$$

$t0 = 1011 0000 0000 0000 0000 0000 0000 0000 →0xB0000000

2.12.4

Yes, it is the desired result. Converting the operands to decimal numbers and subtracting returns a negative number as does the binary subtraction.

2.14

op (6 bits)	rs (5 bits)	rt (5 bits)	rd (5 bits)	shamt (5 bits)	funct (6 bits)
000000	10000	10000	10000	00000	100000
0	$s0	$s0	$s0	0	32

Type = R-type

Assembly = add $s0, $s0, $s0

2.15

op (6 bits)	rs (5 bits)	rt (5 bits)	rd (5 bits)	shamt (5 bits)	funct (6 bits)	address/const
43	\$t2	\$t1	n/a	n/a	n/a	32
101011	01010	01001	n/a	n/a	n/a	0000000000100000

Type = I-type

Binary = 1010 1101 0100 1001 0000 0000 0010 0000

Hex = 0xAD490020

2.17

op (6 bits)	rs (5 bits)	rt (5 bits)	rd (5 bits)	shamt (5 bits)	funct (6 bits)	address/const
0x23	1	2	n/a	n/a	n/a	0x4
35	\$at	\$v0	n/a	n/a	n/a	4
100011	00001	00010	n/a	n/a	n/a	0000000000000100

Type	Instruction	Binary	Hex
I-type	lw \$v0, 4(\$at)	1000 1100 0010 0010 0000 0000 0000 0100	0x8C220004

2.19.2

\$t0 = 1010 1010 1010 1010 1010 1010 1010 1010

sll \$t2, \$t0, 4 → \$t2 = \$t0 << by 4 bits

\$t2 = 1010 1010 1010 1010 1010 1010 1010 0000 = 0xAAAAAAA0

andi \$t2, \$t2, -1 → \$t2 = \$t2 & -1

$$
\begin{array}{cccccccccc}
 & 1010 & 1010 & 1010 & 1010 & 1010 & 1010 & 1010 & 0000 \\
\& & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 \\
\hline
 & 1010 & 1010 & 1010 & 1010 & 1010 & 1010 & 1010 & 0000
\end{array}
$$

\$t2 = 0xAAAAAAA0

2.23

slt \$t2, \$0, \$t0 — $\$0 < \$t0$? **\$t2 = 1** : \$t2 = 0
bne \$t2, \$0, ELSE — go to ELSE if $\$t2 \neq 0$
j DONE — go to DONE
ELSE: addi \$t2, \$t2, 2 — \$t2 = \$t2 + 2
DONE:

\$t2 = 3

2.24

No, you cannot jump the pc from 0x20000000 to 0x40000000 with a jump instruction because a J-type instruction only allows for a jump address of 26 bits.

No, you cannot use beq to jump the pc from 0x20000000 to 0x40000000 because it only allows for an address change of 16 bits.

2.26.1

\$t1 = 10
\$s2 = 0
LOOP: slt \$t2, \$0, \$t1 — $\$0 < \$t1$? \$t2 = 1: \$t2 = 0
beq \$t2, \$0, DONE — go to DONE if \$t2 == 0
subi \$t1, \$t1, 1 — \$t1 = \$t1 - 1
addi \$s2, \$s2, 2 — \$s2 = \$s2 + 2
j LOOP
DONE:

\$t1	9	8	7	6	5	4	3	2	1	0
\$s2	2	4	6	8	10	12	14	16	18	20

Final Value for \$s2 = 20

2.38

\$t1 = 0x1000 0000

\$t2 = 0x1000 0010

Data at 0x1000 000 = 0x11223344

lbu \$t0, 0(\$t1)	\$t0 = 0x11
sw \$t0, 0(\$t2)	Address of \$t2 has the data in \$t0 stored there.

\$t2 has the value 0x00000011

2.40

Address = 0010 0000 0000 0001 0100 1001 0010 0100 = 0x20014924

No, if the PC starts at 0x00000000 it does not have enough bits to change to reach that address.

3.1

5ED4 - 07A4 = 5ED4 + (-07A4)

5ED4 = 0101 1110 1101 0100

07A4 = 0000 0111 1010 0100

inverted 07A4 = 1111 1000 0011 1011

negated 07A4 = 1111 1000 0101 1100

$$
\begin{array}{rrrrr}
 & 0101 & 1110 & 1101 & 0100 \\
+ & 1111 & 1000 & 0101 & 1100 \\
\hline
 & 0101 & 0111 & 0011 & 0000
\end{array}
$$

5ED4 - 07A4 = 5730

3.4

4365 - 3412 = 4365 + (-3412)

4362 = 100 011 110 101

3412 = 011 100 001 010

inverted 3412 = 100 011 110 101

negated 3412 = 100 011 110 110

$$\begin{array}{rrrrr} & 100 & 011 & 110 & 101 \\ + & 100 & 011 & 110 & 110 \\ \hline & 000 & 111 & 101 & 011 \end{array}$$

4365 - 3412 = 0753

3.6

185 - 122 = 185 + (-122)

185 = 10111001

122 = 01111010

inverted 122 = 10000101

negated 122 = 10000110

$$\begin{array}{rr} & 10111001 \\ + & 10000110 \\ \hline & 00111111 \end{array}$$

185 - 122 = 63

Neither overflow or underflow occured.

3.12

62 * 12

62 = 110 010

12 = 001 010

Iteration	Step	Mulitplier	Multiplicand	Product
0	Initial Values	001 010	000 110 010	000 000 000
1	1a: 0 $\Rightarrow$ No operation	001 010	000 110 010	000 000 000
	2: Shift left Multiplicand	001 010	001 100 100	000 000 000
	3: Shift right Multiplier	000 101	001 100 100	000 000 000
2	1a: 1 $\Rightarrow$ Prod = Prod + Mcand	000 101	001 100 100	001 100 100
	2: Shift left Multiplicand	000 101	011 001 000	001 100 100
	3: Shift right Multiplier	000 010	011 001 000	001 100 100
3	1a: 1 $\Rightarrow$ No operation	000 010	011 001 000	001 100 100
	2: Shift left Multiplicand	000 010	110 010 000	001 100 100
	3: Shift right Multiplier	000 001	110 010 000	001 100 100
4	1a: 1 $\Rightarrow$ Prod = Prod + Mcand	000 001	110 010 000	111 110 100
	2: Shift left Multiplicand	000 001	001 100 100 000	111 110 100
	3: Shift right Multiplier	000 000	001 100 100 000	111 110 100

62 * 12 = 111 110 100 = 764