Required Problems:

1 (g), 1 (h), 2 (g)

Show your work for all problems! If you do not show your work, you will not get any credit.

Please include your name and your CS username on your homework (it will make it easier to grade).

Turning It In

You have two options:

- Turn in a hard copy at lecture
- Turn in an electronic copy using turnin; assignment name cs252_f16_hw02

In either case, a typed version (printed out or turned in) is appreciated, but not required.

Allowable Instructions

When writing MIPS assembly, the only instructions that you are allowed to use (so far) are:

- add, addi, sub
- beq, bne, j
- slt, sli
- lw, lh, lb, sw, sh, sb
- la
- syscall

While MIPS has many other useful instructions (and the assembler recognizes many pseudo-instructions), do not use them! We want you to learn the fundamentals of how assembly language works - you can use fancy tricks after this class is over.

Problem 1 - Loading, Storing, and Basic Arithmetic

This question assumes the following MIPS code, which sets up memory locations hermit, kaibab, tanner, clear, creek, ribbon, falls, and tonto. The code then loads the values of some of these variables into the indicated MIPS registers. In answering these questions, you can assume this code has already been executed, and that the value of some of the variables are already in the indicated registers.

Each question is independent of the other questions - that is, assume that the program has started over from scratch each time.
.data
hermit: .word 42
kaibab: .word 912
tanner: .word 8271
clear: .word -879
creek: .word -16
ribbon: .word 2
falls: .word -95
tonto: .word 412

.text
main:
# set $s3 = tonto
la $t0, tonto
lw $s3, 0($t0)

# set $s4 = hermit
la $t0, hermit
lw $s4, 0($t0)

# set $s5 = clear
la $t0, clear
lw $s5, 0($t0)

# set $s6 = creek
la $t0, creek
lw $s6, 0($t0)

(a)
Put clear + creek in register $t9

(b)
Put hermit - creek - clear + tonto in register $t2

(c)
Put hermit + falls in register $t1

(d)
Put tonto - clear + hermit in memory location ribbon

(e)
If ( tonto != hermit ), put tonto + clear in register $s2

(f)
If ( creek >= clear ), put ribbon + clear in register $s2
Problem 2 - Loading, Storing, and Basic Arithmetic

This question assumes the following MIPS code, which sets up the following locations: atsf, dlw, csx, and kcs. The code then loads the values of some of these variables into the MIPS registers. In answering these questions, you can assume this code has already been executed, and that the value of some of the variables are already in the indicated registers.

Each question is independent of the other questions - that is, assume that the program has started over from scratch each time.

.data
atsf: .word 42
dlw: .word 912
bnsf: .word 8271
epsw: .word -879
cbq: .word -16
erie: .word 2
csx: .word -95
kcs .word 412

.text
main:
    # set $t1 = kcs
    la $s0, kcs
    lw $t1, 0($s0)

    # set $t2 = csx
    la $s0, csx
    lw $t2, 0($s0)

    # set $t3 = atsf
    la $s0, atsf
    lw $t3, 0($s0)

    # set $t5 = address of dlw
    la $t5, dlw

(a) Put csx - kcs - atsf in register $t4

(b) Put kcs + csx - atsf in register $s4
(c) Put erie in register $s3

(d) Put dlw - cbq in register $s6

(e) If ( kcs == csx ), put kcs + atsf in register $s2

(f) Put kcs - erie in memory location $csx

(g) - Turn in this one
If ( csx+kcs < erie ), put csx+kcs in register $s3

EXAMPLES

Example: Problem 1(a)

Problem: Put clear + creek in register $t9

```
add $t9, $s5, $s6  # t9 = clear + creek
```

Example: Problem 1(b)

Problem: Put hermit - creek - clear + tonto in register $t2

```
sub $t2, $s4, $s6  # t2 = hermit - creek
sub $t2, $t2, $s5  # t2 = hermit - creek - clear
add $t2, $t2, $s3  # t2 = hermit - creek - clear + tonto
```
Example: Problem 1(c)
Problem: Put `hermit + falls` in register `$t1$

```
la  $t1, falls       # t1 = &falls
lw  $t1, 0($t1)     # t1 = falls
add $t1, $s4, $t1   # t1 = hermit + falls
```

Example: Problem 1(d)
Problem: Put `tonto - clear + hermit` in memory location `ribbon`

```
sub  $t0, $s3, $s5  # t0 = tonto - clear
add $t0, $t0, $s4  # t0 = tonto - clear + hermit
la  $t1, ribbon     # t1 = &ribbon
sw  $t0, 0($t1)     # ribbon = tonto - clear + hermit
```

Example: Problem 1(e)
Problem: If `(tonto != hermit)`, put `tonto + clear` in register `$s2$

```
beq  $s3, $s4, AFTER_IF # if (tonto == hermit) skip ahead
add $s2, $s3, $s5      # if (tonto != hermit) s2 = tonto + clear
```

AFTER_IF:

Example: Problem 1(f)
Problem: If `(creek >= clear)`, put `ribbon + clear` in register `$s2$

```
slt  $t0, $s6, $s5   # t0 = (creek < clear)
bne  $t0, $zero, AFTER_IF # if (creek < clear) skip ahead
la  $t0, ribbon      # t0 = &ribbon
lw  $t0, 0($t0)      # t0 = ribbon
add $s2, $t0, $s5   # s2 = ribbon + clear
```

AFTER_IF:

Example: Problem 2(a)
Put `csx - kcs - atsf` in register `$t4$

```
sub  $t4, $t2, $t1   # t4 = csx - kcs
sub  $t4, $t4, $t3   # t4 = csx - kcs - atsf
```

Example: Problem 2(b)
Put `kcs + csx - atsf` in register `$s4$

```
add  $s4, $t1, $t2   # s4 = kcs + csx
sub  $s4, $s4, $t3   # s4 = kcs + csx - atsf
```
Example: Problem 2(c)
Put \texttt{erie} in register $s3$

\begin{verbatim}
la $s3, \texttt{erie} # s3 = &\texttt{erie}
lw $s3, 0($s3) # s3 = \texttt{erie}
\end{verbatim}

Example: Problem 2(d)
Put \texttt{dlw} - \texttt{cbq} in register $s6$

\begin{verbatim}
lw $s6, 0($t5) # s6 = \texttt{dlw}
lw \texttt{cbq} # t0 = &\texttt{cbq}
lw $t0, 0($t0) # t0 = \texttt{cbq}
sub $s6, $s6, $t0 # s6 = \texttt{dlw} - \texttt{cbq}
\end{verbatim}

Example: Problem 2(e)
If ( \texttt{kcs} == \texttt{csx} ), put \texttt{kcs} + \texttt{atsf} in register $s2$

\begin{verbatim}
bne $t1, $t2, AFTER_IF # if (\texttt{kcs} != \texttt{csx}) skip ahead
add $s2, $t1, $t3 # if (\texttt{kcs} == \texttt{csx}) s2 = \texttt{kcs} + \texttt{csx}
\end{verbatim}

AFTER_IF:

Example: Problem 2(f)
Put \texttt{kcs} - \texttt{erie} in memory location $\texttt{csx}$

\begin{verbatim}
lw $t0, \texttt{erie} # t0 = &\texttt{erie}
lw $t0, 0($t0) # t0 = \texttt{erie}
sub $t0, $t1, $t0 # t0 = \texttt{kcs} - \texttt{erie}
lw $t1, \texttt{csx} # t1 = &\texttt{csx}
sw $t0, 0($t1) # \texttt{csx} = \texttt{kcs} - \texttt{erie}
\end{verbatim}