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, slti
• 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
Put kaibab - (tanner + falls) - tonto in register $t4

Solution:

```assembly
la $t0, tanner       # t0 = &tanner
lw  $t0, 0($t0)      # t0 = tanner
la  $t1, falls       # t1 = &falls
lw  $t1, 0($t1)      # t1 = falls
add $t4, $t0, $t1    # t4 = tanner + falls
la  $t0, kaibab      # t0 = &kaibab
lw  $t0, 0($t0)      # t0 = kaibab
sub $t4, $t0, $t4    # t4 = kaibab - (tanner+falls)
sub $t4, $t4, $s3    # t4 = kaibab - (tanner+falls) - tonto
```

If (hermit <= tonto), put hermit - tonto in memory location creek.

Solution:

```assembly
slt  $t0, $s3, $s4    # t0 = (tonto < hermit)
bne $t0, $zero, AFTER_IF # if (tonto < hermit) jump ahead
sub $t0, $s4, $s3    # t0 = hermit - tonto
la  $t1, creek       # t1 = &creek
sw  $t0, 0($t1)      # creek = hermit - tonto
```

AFTER_IF:

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

Solution:
  add $t0, $t2, $t1  # t0 = csx+kcs
  la  $t1, erie      # t1 = &erie
  lw  $t1, 0($t1)    # t1 = erie
  slt $t1, $t0, $t1  # t1 = (csx+kcs) < erie
  beq $t1, $zero, AFTER_IF # if (csx+kcs >= erie) jump ahead
  add $s3, $t0, $zero
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 \((\text{tonto} \neq \text{hermit})\), put \text{tonto} + \text{clear} in register \$s2

\begin{align*}
\text{beq} \quad &\$s3, \$s4, \text{AFTER\_IF} \quad # \text{if (tonto} = \text{hermit) skip ahead} \\
\text{add} \quad &\$s2, \$s3, \$s5 \quad # \text{if (tonto} \neq \text{hermit) s2} = \text{tonto} + \text{clear}
\end{align*}

AFTER\_IF:

Example: Problem 1(f)
Problem: If \((\text{creek} \geq \text{clear})\), put \text{ribbon} + \text{clear} in register \$s2

\begin{align*}
\text{slt} \quad &\$t0, \$s6, \$s5 \quad # \text{t0} = (\text{creek} < \text{clear}) \\
\text{bne} \quad &\$t0, \$zero, \text{AFTER\_IF} \quad # \text{if (creek} < \text{clear) skip ahead} \\
\text{la} \quad &\$t0, \text{ribbon} \quad # \text{t0} = &\text{ribbon} \\
\text{lw} \quad &\$t0, 0(\$t0) \quad # \text{t0} = \text{ribbon} \\
\text{add} \quad &\$s2, \$t0, \$s5 \quad # \text{s2} = \text{ribbon} + \text{clear}
\end{align*}

AFTER\_IF:

Example: Problem 2(a)
Put \text{csx} - \text{kcs} - \text{atsf} in register \$t4

\begin{align*}
\text{sub} \quad &\$t4, \$t2, \$t1 \quad # \text{t4} = \text{csx} - \text{kcs} \\
\text{sub} \quad &\$t4, \$t4, \$t3 \quad # \text{t4} = \text{csx} - \text{kcs} - \text{atsf}
\end{align*}

Example: Problem 2(b)
Put \text{kcs} + \text{csx} - \text{atsf} in register \$s4

\begin{align*}
\text{add} \quad &\$s4, \$t1, \$t2 \quad # \text{s4} = \text{kcs} + \text{csx} \\
\text{sub} \quad &\$s4, \$s4, \$t3 \quad # \text{s4} = \text{kcs} + \text{csx} - \text{atsf}
\end{align*}

Example: Problem 2(c)
Put \text{erie} in register \$s3

\begin{align*}
\text{la} \quad &\$s3, \text{erie} \quad # \text{s3} = &\text{erie} \\
\text{lw} \quad &\$s3, 0(\$s3) \quad # \text{s3} = \text{erie}
\end{align*}

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

\begin{align*}
\text{lw} \quad &\$s6, 0(\$t5) \quad # \text{s6} = \text{dlw} \\
\text{la} \quad &\$t0, \text{cbq} \quad # \text{t0} = &\text{cbq} \\
\text{lw} \quad &\$t0, 0(\$t0) \quad # \text{t0} = \text{cbq} \\
\text{sub} \quad &\$s6, \$s6, \$t0 \quad # \text{s6} = \text{dlw} - \text{cbq}
\end{align*}
Example: Problem 2(e)

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

    bne  $t1, $t2, AFTER_IF  # if (kcs != csx) skip ahead
    add  $s2, $t1, $t3       # if (kcs == csx) s2 = kcs + csx

AFTER_IF:

Example: Problem 2(f)

Put kcs - erie in memory location $csx

    la   $t0, erie          # t0 = &erie
    lw   $t0, 0($t0)        # t0 = erie
    sub  $t0, $t1, $t0      # t0 = kcs - erie
    la   $t1, csx           # t1 = &csx
    sw   $t0, 0($t1)        # csx = kcs - erie