EECS 370
Homework #1 - SOLUTIONS

Problem #1

We need to multiply r1 by 100.
There are many ways to do this with a sequence of 8 adds.
Here is one of them. For the purpose of the comments,
we will call the original value in r1 "x".

```
  add 2 1 1    r2 = x * 2
  add 2 2 2    r2 = x * 4
  add 2 2 1    r2 = x * 5
  add 1 2 2    r1 = x * 10
  add 2 1 1    r2 = x * 20
  add 2 2 2    r2 = x * 40
  add 2 2 1    r2 = x * 50
  add 2 2 2    r2 = x * 100
```

Problem #2

227 instructions:

```
  2 initial instructions
 25 loop iterations of 4 instructions (internal branch taken)
 25 loop iterations of 5 instructions (internal branch not taken)
```

Code description:

```
sum = 0
for (i = 50; i != 0; i--) {
  if (i & 1) {
    sum = sum + i
  }
}
```

Text explanation:

This computes the sum of the odd numbers in the range 1 through 50.

Problem #3

```
while (save[i] == k)
    i = i + j;
```

Original code:

```
Loop:   add $t1,$s3,$s3    # Temp reg $t1 = 2 * i
        add $t1,$t1,$t1    # Temp reg $t1 = 4 * i
        add $t1,$t1,$s6    # $t1 = address of save[i]
        lw  $t0,0($t1)     # Temp reg $t0 = save[i]
        bne $t0,$s5,Exit   # go to Exit if save[i] != k
        add $s3,$s3,$s4    # i = i + j
        j   Loop           # go to Loop
Exit:
```

Optimized code:

```
        j   Enter          # Jump into loop entry point
Loop:   add $s3,$s3,$s4    # i = i + j
Enter:  sll $t1,$s3,2      # Temp reg $t1 = 4 * i
        add $t1,$t1,$s6    # $t1 = address of save[i]
        lw  $t0,0($t1)     # Temp reg $t0 = save[i]
        beq $t0,$s5,Loop   # go to Exit if save[i] != k

Exit:
```

Problem #4

Textbook problem 3.19, page 203
(Refer to the "In More Depth" section on pages 201-202)

Assumptions:
- Opcode: 1 byte
- Register: 1/2 byte
- Memory address: 2 bytes
- Data operand: 4 bytes
- All instructions integral number of bytes

Example on page 202:
```
a = b + c;
```

Accumulator:

	Code	Data	Total
load b	3	4	7
add c	3	4	7
store a	3	4	7

Instructions: 3
Code size: 9 bytes
Memory transferred: 12 bytes
Total memory bandwidth: 21 bytes

Memory-memory:

	Code	Data	Total
add a b c	7	12	19

Instructions: 1
Code size: 7 bytes
Memory transferred: 12 bytes
Total memory bandwidth: 19 bytes

Stack:

	Code	Data	Total
push b	3	4	7
push c	3	4	7
add	1	0	1
pop a	3	4	7

Instructions: 4
Code size: 10 bytes
Memory transferred: 12 bytes
Total memory bandwidth: 22 bytes

Load-store:

	Code	Data	Total
load r1 a	4	4	8
load r2 b	4	4	8
add r3 r2 r1	3	0	3
store r3 c	4	4	8

Instructions: 4
Code size: 15 bytes
Memory transferred: 12 bytes
Total memory bandwidth: 27 bytes

Problem 3.19 Program:

```
a = b + c;
b = a + c;
d = a - b;
```

Accumulator, without reversed sub:

	Code	Data	Total
load b	3	4	7
add c	3	4	7
store a	3	4	7
add c	3	4	7
store b	3	4	7
load d	3	4	7
sub b	3	4	7
store d	3	4	7

Instructions: 8
Code size: 24 bytes
Memory transferred: 32 bytes
Total memory bandwidth: 56 bytes

Accumulator, with reversed sub:

	Code	Data	Total
load b	3	4	7
add c	3	4	7
store a	3	4	7
add c	3	4	7
store b	3	4	7
subr d	3	4	7
store d	3	4	7

Instructions: 7
Code size: 21 bytes
Memory transferred: 28 bytes
Total memory bandwidth: 49 bytes

Memory-memory:

	Code	Data	Total
add a b c	7	12	19
add b a c	7	12	19
sub d a b	7	12	19

```
  Instructions:  3
  Code size:     21 bytes
  Memory transferred:  36 bytes
  Total memory bandwidth:  57 bytes

  a = b + c;
  b = a + c;
  d = a - b;

Stack:
              Code    Data  Total
  push b        3       4     7
  push c        3       4     7
  add           1       0     1
  pop  a        3       4     7
  push a        3       4     7
  push c        3       4     7
  add           1       0     1
  pop  b        3       4     7
  push a        3       4     7
  push b        3       4     7
  sub           1       0     1
  pop  d        3       4     7

  Instructions:  12
  Code size:     30 bytes
  Memory transferred:  36 bytes
  Total memory bandwidth:  66 bytes

Load-store:
              Code    Data  Total
  load r1 a     4       4     8
  load r2 b     4       4     8
  add r3 r2 r1  3       0     3
  store r3 c    4       4     8
  add r2 r1 r3  3       0     3
  store r3 b    4       4     8
  sub r4 r1 r2  3       0     3
  store r4 d    4       4     8

  Instructions:  8
  Code size:     29 bytes
  Memory transferred:  20 bytes
  Total memory bandwidth:  49 bytes

Summary:

                 Instruction    Memory         Total
                    bytes     data bytes      memory
                   fetched    transferred    bandwidth

Accumulator          21           28            49
Memory-memory        21           36            57
Stack                30           36            66
Load-store           29           20            49
```

Problem #5

```
a  Static/Global
b  Static/Global
c  Stack
d  Stack
e  Static/Global
f  Stack
g  Stack
h  Stack
```