Peter Woodman
0127119
CSE 410
H. Levy
Homework #1

2. The following code returns how many times 2 goes into the input.

```
begin:  addi $t0, $zero, 0      # initialize couner
        addi $t1, $zero, 1      # initialize
loop:   slt $t2, $a0, $t1       # if input less than test
        bne $t2, $zero, finish  # 2 divides $a0 $t0 times
        add $t0, $t0, $t1       # otherwise
        addi $t1, $t1, 2        # check next multiple of two
        j loop                  #
finish: add $v0, $t0, $zero     # return result
```

3. This program doubles the values of a continuous block of words in memory, starting at $\texttt{\$s0} + 4 \cdot \texttt{\$2}$ and ending at $\texttt{\$s0} + 4 \cdot (\texttt{\$s1} - 1)$. Charts of memory and registers at each loop step of the example state are reproduced below.

Address	start	1	2	3	4	5
400	11	22	22	22	22	22
404	23	23	46	46	46	46
408	63	63	63	126	126	126
412	128	128	128	128	256	256
416	911	911	911	911	911	1822

Register	start	1	2	3	4	5
`$s0`	400	400	400	400	400	400
`$s1`	5	5	5	5	5	5
`$s2`	0	1	2	3	4	5
`$t0`	$\emptyset$	400	404	408	412	416
`$t1`	$\emptyset$	22	46	126	256	1822

Extra Credit: Optimized code of the above routine is included below. I changed the mult by 4 part of the code from two adds to one shift, and eliminated the unconditional jump. This code now takes two less instructions per loop- the multiply by 4 takes one less instruction, and the constant jump has been removed. `beq` and `bne` are both pseudoinstructions taking two real instructions, so the original took 10 instructions per loop where this one takes 8 - a 20% gain in this function!

```
start:  sll     $t0, $s2, 2
        add     $t0, $t0, $s0
        lw      $t1, 0($t0)
        add     $t1, $t1, $t1
        sw      $t1, 0($t0)
        addi    $s2, $s2, 1
        bne     $s1, $s2, start
```

4. (a) $393 = \texttt{0000 0000 0000 0000 0000 0001 1000 1001} = \texttt{0x0000189}$
   (b) $-3383 = \texttt{1111 1111 1111 1111 1111 0010 1100 1001} = \texttt{0xFFFFF2C9}$
5. (a) $\texttt{1111 1111 1111 1111 1111 1101 1100 1101} = \texttt{0xFFFFFDCD} = -562$
   (b) $\texttt{0011 1111 1001 1111 0000 1000 0011 0101} = \texttt{0x3F9F0835} = 1,067,386,933$
6. (a) $\texttt{0x11001001} = \texttt{0001 0001 0000 0000 0001 0000 0000 0001}$
   (b) $\texttt{0xDEADB33F} = \texttt{1101 1110 0110 1101 0111 0011 0011 1111} = -563,252,417$

7. I hope the code below is clear- the entry point should be `loop`, at the top.

```
loop:   addi $v0, $zero, 0
        j    loop1
loop2:  addi $v0, $v0, 1
        addi $a0, $a0, 4
        addi $a1, $a1, 4
loop1:  lw   $v1, 0($a0)
        sw   $v1, 0($a1)
        bne  $v1, $zero, loop2
```

8.
```
lui  $t1, 15625     # i am unsure how you expect us to reference
srl  $t1, $t1, 8    # the array x, so, i am explicitly setting it.
lw   $t2, 44($t1)   # $t2 = x[11]
add  $t2, $t2, $t0  # $t2 = x[11] + c
sw   $t2, 40($t1)   # x[10] = x[11] + c
```

9.
```
add  $t1, t2, $t3  # = 0000 0001 0100 1011 0100 1000 0010 0000 = 0x014A4820
addi $s0, $t1, 3   # = 0010 0001 0011 0000 0000 0000 0000 0011 = 0x21300003
addi $s0, $t1, -3  # = 0010 0001 0011 0000 1111 1111 1111 1101 = 0x2130FFFD
sw   $t3, 32($t2)  # = 0110 1101 0100 1011 0000 0000 0010 0000 = 0x6D4B0020
```