CS64 homework #2, Due Monday, April 25, 4:30pm

1. (10pt) Add comments to the following MIPS code and describe in one sentence what it does. Assume that $s0 is used for the input and initially contains the number 100. Assume that $v0 is used for the output

```
begin:  add $t0, $0, $0
loop:   beq $s0,$0,finish
        add $t0, $t0, $s0
        sub $s0,$s0, 2
        j loop
finish: add $v0,$t0, $t0
```

2. (10pt) Why doesn’t MIPS have a subtract immediate (subi) instruction? How would you convert the instruction subi $s0, $s1, 0xFF00 to an equivalent instruction using addi? Recall that MIPS uses the 2’s complement number format, and in arithmetic instructions, the immediate is always sign-extended.

3. (40pt) As discussed in class, pseudo-instructions are not part of the native MIPS instruction set but often appear in MIPS programs. We arbitrarily build a few pseudo instructions in the following table. You need to produce a minimal sequence of native (actual) MIPS instructions to accomplish the same thing. **NOTE:** You are not allowed to use any registers other than the one included in the instructions, except register $at which is reserved for assembler and the zero register $0.

```
a. move $a1,$a2  # $a1=$a2
b. neg $a1,$a2  # $a1=-$a2
c. clear $a0    # $a0=0
d. pow $a1, n   # $a1=$a1*n, where n is a multiple of 2.
e. ble $a1, $a2, LABEL  # if $a1<=$a2, jump to LABEL
```

4. (40pt) Implement the following C code in MIPS:

```
for (i=0; i < 4; i++) {
    a[i] = b[2i] + 1;
}
```

Assume that the base address at b[] (b is an array of 10 integers) is stored in $s0, and the base address of a[] (a is an array of 5 integers) is stored in $s1. At the end of the program, print the content of a[]. Write your MIPS code by modifying the following program, make sure to include detailed comments for each instruction. Submit your code (include the whole program) using turnin, under the name hw2. We will test the whole program under QtSPIM.
.data
Array_b:.word 1 2 3 4 5 6 7 8 9 10  # declare an array of 10 integers
Array_a:.word 1 2 3 4 5  # declare an array of 5 integers

.text
main:  la  $s0, Array_b   # load the address of b into $s0
   la  $s1, Array_a   # load the address of a into $s1

your program starts from here

:
:
:
la  $s1, Array_a
Print: lw $a0, 0 ($s1)
   li  $v0,1  # Set up system call for printing integer
   syscall  # Print the value stored in $a0 to screen
:
:

your program as well
:
li $v0, 10  # syscall code 10 for terminating the program
syscall