**1.** [5points] Prior to the early 1980s, machines were built with more and more complex instructionset. The MIPS is a RISC machine. Why has there been a move to RISC machines away from complex instruction machines?

There are number of reasons for the move towards RISC machines away from CISC. Some of them are:

* + Since early computers had limited memory capacities and were expensive, having a CISC instruction set enabled performing complex operations with very few instructions (encoded within a smaller memory size compared to a corresponding RISC program). Since then memories have got cheaper and there has been a lot of advances in the design of cache hierarchies (for example, a dedicated on-chip instruction cache, prefetching techniques, etc.) that permit RISC machines work-around longer instruction sequences
  + Writing a compiler to generate efficient code is easier for a RISC architecture than for a CISC architecture as the compiler can take advantage of a lot of registers provided by the RISC architecture than a CISC
  + RISC instructions are easier to pipeline than CISC instructions

1. [5 points] Write the following sequence of code into MIPS assembler:

x=x-y+z-q;

Assume that x,y,z,q are stored in registers $s1-$s4.

A: The MIPS assembly sequence is as follows:

Sub $t0,$s1,$s2

Add $t1,$t0,$s3

Sub $s1,$t1,$s4

2.[6points] Some machines have a special flag register which contains status bits. These bits ofteninclude the *carry* and *overflow* bits. Describe the difference between the functionality of these two bits and give an example of an arithmetic operation that would lead to them being set to different values.

The *carry* flag is set when arithmetic operation results in generating a carry bit out of the most significant bit position of its operand. The *overflow* flag is set when an arithmetic operation results in generating a carry bit out of the most significant bit position of the physical register which holds its operand. An overflow means that the register size is not big enough to hold the result of the current arithmetic operation while a carry just indicates that the resulting value’s most significant bit position is higher (or lower in case of a borrow) than its operand by a bit position. When we add two integers: 0x0100 and 0x0110 which are held in 16-bit registers, the result 0x1010 generates a carry but not a overflow bit.

**5.** [3points] The MIPS instruction set includes several shift instructions. They include logical-shift-left, logical-shift-right, and arithmetic-shift-right.

a) Why doesn’t MIPS offer an “arithmetic-shift-left” opcode?

The logical and arithmetic left shift operations are the same. That is why there is no need for a separate arithmetic left shift operation.

6. (b) Assuming single precision IEEE 754 format, what decimal number is represent by this word:

1 01111101 00100000000000000000000

(Hint: remember to use the biased form of the exponent.)

The decimal number

* + (+1)\* (2^(125-127))\*(1.001)2
  + (+1)\*(0.25)\*(0.125)
  + 0.03125

1. [12 points] Perform the following operations by converting the operands to 2’s complement binary numbers and then doing the addition or subtraction shown. Please show all work in binary, operating on 16-bit numbers.

(a) 3 + 12

0000 0000 0000 0011 (3)

* 0000 0000 0000 1100 (12)

0000 0000 0000 1111

The decimal value of the result is 15

(b) 13 – 2

0000 0000 0000 1101 (13)

* 1111 1111 1111 1110 (-2)

0000 0000 0000 1011

The decimal value of the result is 11 (we ignored the carry here)

(c) 5 – 6

0000 0000 0000 0101 (5)

* 1111 1111 1111 1010 (-6)

1111 1111 1111 1111

The decimal value of the result is -1

8. [5points]Given the following MIPS assembly code (and assuming all registers start at 0):

addi $1 $0 3

addi $2 $0 8

loop add $1 $1 $2

and $3 $1 $2

addi $2 $2 -2

bne $0 $2 loop

9.