Write each of the programs in this homework and run them on your SPIM emulator. DO NOT just write something in the area to the right. COMPOSE THE PROGRAMS ACCORDING TO THE DIRECTIONS AND MAKE SURE EACH RUNS PROPERLY AND GIVES THE CORRECT RESULT! Then turn in the programs when the homework is due.

1. (CLO 5—Assy Lang.) Write a program to do the following: Using the li instruction, load 8 in $t3, 31 in $t4, and 14 in $t5. Multiply the contents of $t3 and $t4, putting the result in $t0. AND the contents of $t4 and $t5, putting the result in $t1. OR the contents of $t1 from $t0, putting the result in $a0. Output the contents of $a0, $t0, and $t1. Note that no data declaration is necessary. Write down the value of $a0 on your answer sheet. Remember to end your program correctly. Output each answer of the three on a different line (i.e., do a CR/LF between each answer).

Remember: You can only perform mathematical operations on data that is in registers.

2. (CLO 5—Assy Lang.) Write a program to do the following: OR the words t and u together and store the result in x. XOR v and w, storing the result in y. Then add x and y and store the result in z. Multiply x times y times z and output that result to the console.

Remember: You can only perform mathematical operations on data that is in registers.
3. (CLO 5—Assy Lang.) The data declarations for the program below are done. Remembering that data can only be manipulated in registers, subtract data2 from data1. Then add that result to data3. Multiply that result by data4, then OR the resulting product with data 5. Finally, take the 2’s complement of the result, store this final result in ans, and output it to the console. Before you output the answer, output the leader declared below: “Answer = ”. Then output the answer you have calculated.

```
.data
str: .asciiz "Answer = 
data1: .word 40
data2: .word 30
data3: .word 20
data4: .word 10
data5: .word 5
ans: .word 0
```
4. (CLO 5—Assy Lang.) We have not studied branch instructions yet, but they are actually quite easy to understand. In a branch instruction, a test is made for a certain condition, normally comparing two registers or perhaps determining the value of a single register. An example is bgez $t1,next. This branch instruction instructs the computer to evaluate the contents of the register $t1. If the contents are greater than or equal to 0 ($\geq 0$), then the computer immediately goes to the instruction labeled “next,” wherever it is in the program, and executes it, continuing to execute instructions from that point. If the contents of $t1$ are less than zero, the computer simply executes the instruction that follows the branch instruction.

Let’s use that simple branch instruction to do the following: Load each of the two words declared in the data statement shown to the right into registers, and then determine if either is $\geq 0$. If either number is $\geq 0$, print it out using syscall 1. If a number is negative, do not print it out. End the program with a syscall 10. Note: you can declare words as either decimal or hexadecimal. SPIM understands both number systems. Hint: the branch instruction that will help is bltz (branch on less than 0 [$<0$]). An example would be:

```
bltz $t0,next  # “If $t0 is less than 0, go to the instruction labeled ‘next;’ otherwise do the next instruction.”
```

If only one number is printed out, which number is it?

```
.text
main:

.data
data1: .word 0x63f2e3
data2: .word 0x9100ab72
```
5. (CLO 5—Assy Lang.) In a data statement, declare six words, aa, bb, cc, dd, ee, and ff. The words cc-ff are placeholders; declare each with a value of 0. The value of aa is 1408, the value of bb is 987. In the program subtract bb from aa and store in cc. Multiply bb times cc and store the results in dd. OR cc and dd and store results in ee. AND dd and ee and store that final result in ff. Output only the value of ff, with a leader that says, “Final answer = ”. Stop the program as usual.

```
.text
main:
```