Assembler and Circuits
Due: April 6, 2018

QUESTION 1: ASSEMBLER PROGRAMMING

Convert the following C program into MIPS and get it to run on the MARS emulator:

```
int array[10]; // uninitialized array

main() {
  int i, sum=0;
  for(i=0; i<10; i++) {
     printf("enter a number: ");
     scanf("%d",&array[i]);
  }
  sum=summation(array, array+10);
  printf("the sum is %d”,sum);
}

int summation(int *a, *last) {
   int result=0;
   if (a != last) result=summation(a+1, last)+*a;
   return result;
}
```

Submit the source file to myCourses. The TA will compile and run it.

QUESTION 2: CIRCUIT DRAWING

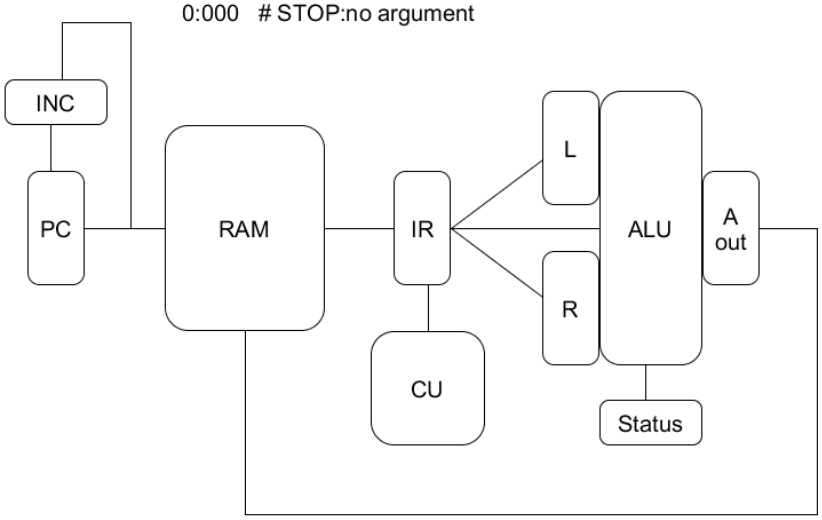
Using your RAM from assignment 1, your ALU from assignment 2, merge them into a single circuit according to the following pattern:

The only assembler instructions:

```
ADD constant  # AOUT = constant + constant
MOV address   # [address] = AOUT
STOP          # Clock tick does not effect circuit
```

Machine language: OP:ARG

```
0:NNN  # ADD:the number to add
1:ZZZ  # MOV:the address in RAM to store
0:000  # STOP:no argument
```



Step 1: RAM is initialized with 2 machine language instructions.
Step 2: PC is initialized with 0.
Step 3: Clock starts ticking and CU begins its fetch cycle (not shown)
Step 4: IR = RAM[PC] and PC = INC(PC)
Step 5: CU = OP-Code
Step 6: IR + CU causes either the ADD or MOV operation to execute, or skip
Step 7: If IR + CU is all zeros then STOP
Step 8: If not STOP then go to step 4

Using Logisim construct the above circuit template. Your 4 nibble RAM will be populated by any combination of the two instructions ADD or MOV stored in addresses 0 and 1. Address 2 will optionally have the STOP instruction. Address 3 in your RAM will optionally store the result. Notice that the ideal algorithm would be two instructions followed by a stop and one space to store a result, but this is optional. That means your circuit must be able to handle other cases. For example, the circuit can store results in any address, overwriting what is already present. For example, if there is no STOP then it would cycle back to address 0 and continue executing in an infinite loop. The Status register stores these flags: zero, arithmetic overflow.

WHAT TO HAND IN

Question 1 – the source file for the TA to execute.

Question 2 – the logisim circuit file for the TA to execute.

ZIP the files of your submission.

HOW IT WILL BE GRADED

Question 1: 10 points
Question 2: 10 points
Total points = 20

Each question is graded proportionally to how correct your solution is compared with the official solution.