Question 1: (10 points)

The intention of the subroutine `multN` is to multiply two numbers, each formed by `N` bits for $0 < \texttt{N} \leq 16$. It expects the $(32-\texttt{N})$ most-significant bits of the operands of the multiplication to be all zero. If there are any non-zero bits amongst these most-significant bits of the operands, then the subroutine returns 1 in `$v1`. Otherwise it returns 0 in `$v1` and returns the value of the product in `$v0`.

A control flow graph (CFG) is a graph commonly generated by compilers to analyze the flow of control of a program. Figure 1 shows the CFG for the `multN` subroutine. Each node in the CFG is a basic block and an edge in the CFG indicates that it is possible for the execution of the program to flow from the last instruction in the basic block that is the source of the edge to the first instruction in the basic block that is the target of the edge.

a. (5 points) The arguments for `multN` are `N`, the `multiplier` and the `multiplicand`. Examining the code above, indicate the register that contains each argument.

Argument	Register
`N`	
`multiplicand`	
`multiplier`	

b. (5 points) In the table below indicate how many times the statements in lines `09` and `11` are executed when `multN` is executed with the given arguments.

Arguments			Number of Executions	
`N`	`multiplicand`	`multiplier`	line `09`	line `11`
16	`0x0000 000C`	`0x0000 0003`		
8	`0x0000 000C`	`0x0000 0003`		
16	`0x0000 FFFF`	`0x0000 8000`		
16	`0x0000 8000`	`0x0000 FFFF`		
4	`0x0000 0808`	`0x0000 0002`		

```
00 multN: addi  $v1, $zero, 1
01        bgt   $a2, 16, done
02        beqz  $a2, done
03        srlv  $t2, $a0, $a2
04        bne   $t2, $zero, done
05        srlv  $t3, $a1, $a2
06        bne   $t3, $zero, done
07        move  $v1, $zero
08        move  $v0, $zero
09 next:  andi  $t1, $a1, 0x0001
10        beqz   $t1, shift
11        add   $v0, $v0, $a0
12 shift: sll   $a0, $a0, 1
13        srl   $a1, $a1, 1
14        subi  $a2, $a2, 1
15        bne   $a2, $zero, next
16 done:  jr    $ra
```

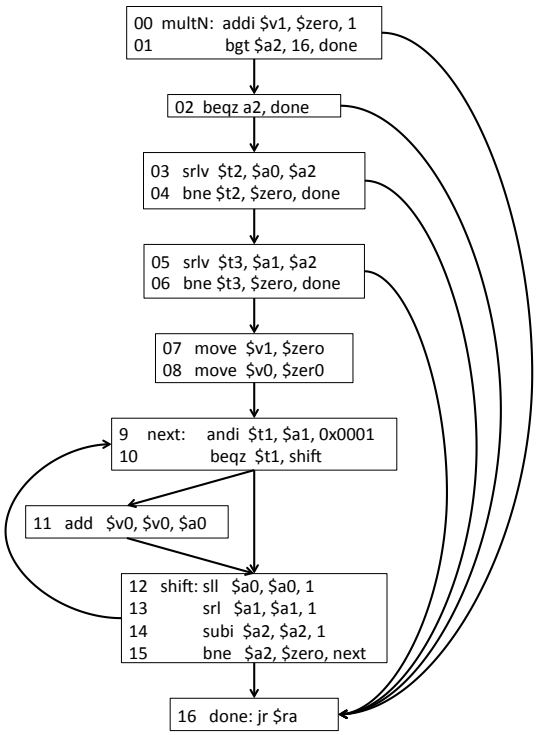

Figure 1: Control Flow Graph for `multN`.