```
  .arch armv6
  .fpu vfp
  .text

@ print function is complete, no modifications needed
   .global print
print:
  stmfd sp!, {r3, lr}
  mov   r3, r0
  mov   r2, r1
  ldr   r0, startstring
  mov   r1, r3
  bl printf
  ldmfd sp!, {r3, pc}

startstring:
  .word string0

   .global towers
towers:

  sub sp, sp, #4
  str lr, [sp, #0]

  /* Save calllee-saved registers to stack */
  @assume r4 = steps, r5 = peg

  /*push {r4, r5}*/

  /* Save a copy of all 3 incoming parameters */
  @ assume r6 = numDisks, r7 = start, r8 = goal

  push {r4, r5, r6, r7, r8}

  mov r6, r0
  mov r7, r1
  mov r8, r2

if:
  /* Compare numDisks with 2 or (numDisks - 2)*/
  cmp r0, #2
  /* Check if less than, else branch to else */
  bge else
  /* set print function's start to incoming start */
  mov r0, r1
  /* set print function's end to goal */
```

```
  mov r1, r2
  /* call print function */
  bl print
  /* Set return register to 1 */
  mov r0, #1
  /* branch to endif */
  bl endif

else:
  /* Use a callee-saved varable for temp and set it to 6 */
  mov r5, #6
  /* Subract start from temp and store to itself */
  sub r5, r5, r1
  /* Subtract goal from temp and store to itself (temp = 6 - start - goal)*/
  sub r5, r5, r2
  /* subtract 1 from original numDisks and store it to numDisks parameter */
  sub r0, r0, #1
  /* Set end parameter as temp */
  mov r2, r5
  /* Call towers function */
  bl towers
  /* Save result to callee-saved register for total steps */
  mov r4, r0
  /* Set numDiscs parameter to 1 */
  mov r0, #1
  /* Set start parameter to original start */
  mov r1, r7
  /* Set goal parameter to original goal */
  mov r2, r8
  /* Call towers function */
  bl towers
  /* Add result to total steps so far */
  add r4, r4, r0
  /* Set numDisks parameter to original numDisks - 1 */
  sub r0, r6, #1
  /* set start parameter to temp */
  mov r1, r5
  /* set goal parameter to original goal */
  mov r2, r8
  /* Call towers function */
  bl towers
  /* Add result to total steps so far and save it to return register */
  add r0, r4, r0

endif:
  /* Restore Registers */
```

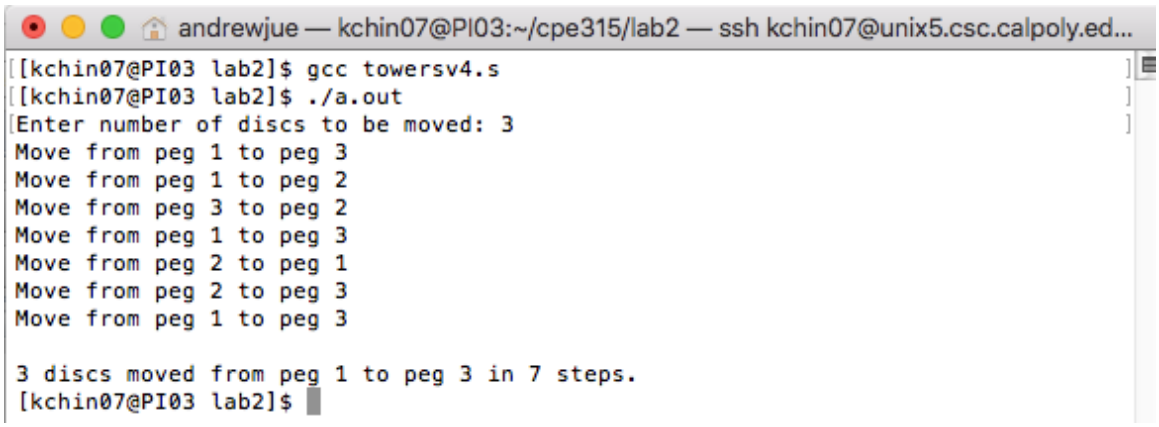
```
  pop {r4, r5, r6, r7, r8}

  add sp, sp, #4
  ldr pc, [sp, #-4]

@ Function main is complete, no modifications needed
   .global main
main:
  str   lr, [sp, #-4]!
  sub   sp, sp, #20
  ldr   r0, printdata
  bl printf
  ldr   r0, printdata+4
  add   r1, sp, #12
  bl scanf
  ldr   r0, [sp, #12]
  mov   r1, #1
  mov   r2, #3
  bl towers
  str   r0, [sp]
  ldr   r0, printdata+8
  ldr   r1, [sp, #12]
  mov   r2, #1
  mov   r3, #3
  bl printf
  mov   r0, #0
  add   sp, sp, #20
  ldr   pc, [sp], #4
end:

printdata:
  .word string1
  .word string2
  .word string3

string0:
  .asciz   "Move from peg %d to peg %d\n"
string1:
  .asciz   "Enter number of discs to be moved: "
string2:
  .asciz   "%d"
  .space   1
string3:
  .ascii   "\n%d discs moved from peg %d to peg %d in %d steps."
  .ascii   "\012\000"
```


andrewjue — kchin07@PI03:~/cpe315/lab2 — ssh kchin07@unix5.csc.calpoly.ed...
[kchin07@PI03 lab2]$ gcc towersv4.s
[kchin07@PI03 lab2]$ ./a.out
Enter number of discs to be moved: 3
Move from peg 1 to peg 3
Move from peg 1 to peg 2
Move from peg 3 to peg 2
Move from peg 1 to peg 3
Move from peg 2 to peg 1
Move from peg 2 to peg 3
Move from peg 1 to peg 3
3 discs moved from peg 1 to peg 3 in 7 steps.
[kchin07@PI03 lab2]$