CompArch ECGR 4181/5181 Question

1. Consider a function to calculate the factorial of a positive integer (shown below).

```c
#include<stdio.h>
long int multiplyNumbers(int n);
int main() {
    int n;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    printf("Factorial of %d = %ld", n, multiplyNumbers(n));
    return 0;
}

long int multiplyNumbers(int n) {
    if (n>=1)
        return n*multiplyNumbers(n-1);
    else
        return 1;
}
```

   a) Write the RISC-V assembly codes for both the functions *main* and *multiplyNumbers* with the assumption that data and instruction words are 32 bits wide, memory is byte addressable, and Stack Pointer (SP) will have to be used. Make assumptions as you see fit, but state and explain them clearly. The control of execution once *multiplyNumbers* ends, will have to move back to the function *main*. [25]

   b) Draw a diagram of the control of flow between the different registers and memory locations, and update their contents/addresses during the step by step of execution of the code. Assume instruction word addresses of your choice and point out the contents of the PC (program counter) and SP for every line of assembly code. [10]

2. For this question, see the state machine on the final page of this test: [15 + 20]

(a)Fill out the following chart for a two-processor system executing the following sequence of instructions and adhering to the cache protocol in the diagram. Assume the following:

- The data from addresses 110 and 120 are initially stored in both processors' caches and marked as clean/shared.
- The cache block size is 1 word.

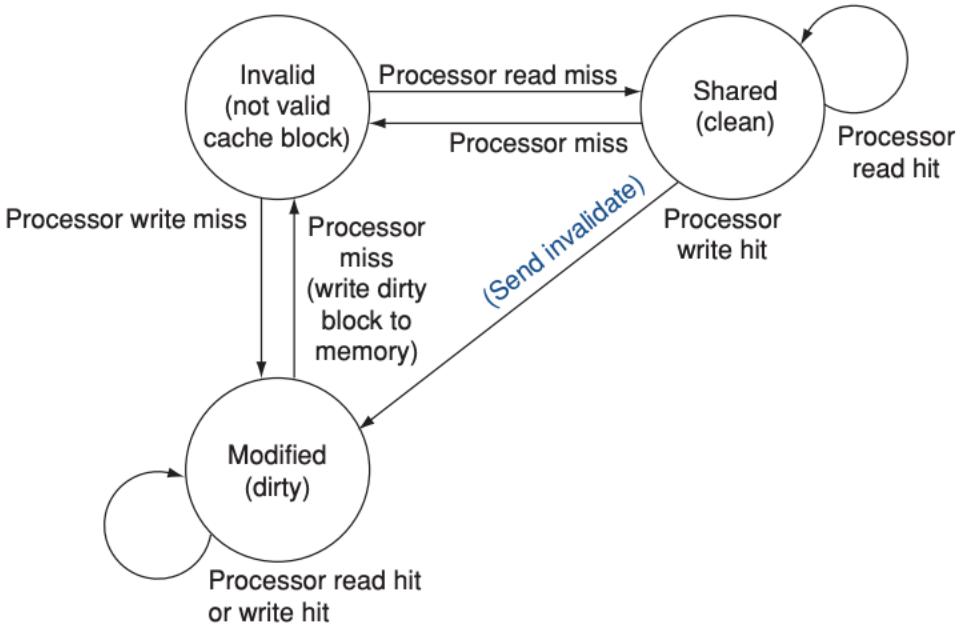
"Before" and "After" refer to the block's state in cache before the instruction is executed and after it is executed, respectively.

Processor: Instruction	**Hit or miss?**	**Block's state in P0 cache (before)**	**Block's state in P0 cache (after)**	**Block's state in P1 cache (before)**	**Block's state in P1 cache (after)**
P0: read 120					
P0: write 120					
P1: write 120					
P1: read 110					
P0: write 110					
P1: read 120					

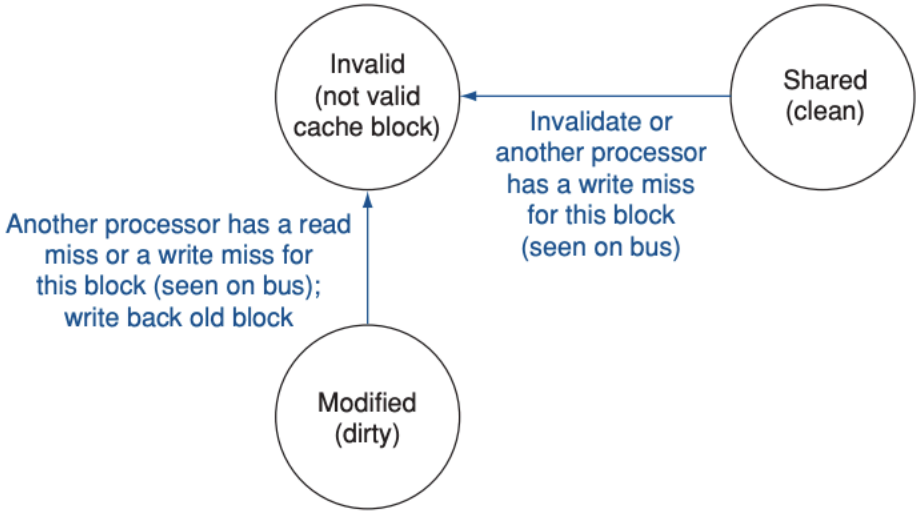
(b) The MESI protocol is similar to the protocol in the diagram, except that it has an additional state called "Exclusive." If a block is marked "Exclusive," that means that it is present in only one processor's cache and not shared in any other processor's cache, and that it is clean (has not been written). What is the advantage of this additional state?

Fill the table below for the MESI protocol (using assumptions from part (a) above):

Processor: Instruction	**Hit or miss?**	**Block's state in P0 cache (before)**	**Block's state in P0 cache (after)**	**Block's state in P1 cache (before)**	**Block's state in P1 cache (after)**
P0: read 120					
P0: write 120					
P1: write 120					
P1: read 110					
P0: write 110					
P1: read 120					



a. Cache state transitions using signals from the processor



b. Cache state transitions using signals from the bus