Arch Lab 2

Iris Taubkin 208410969 Omri Elad 204620702

1.
   1.1. Suppose at the current cycle, r[2] = 12, r[3] = 45, r[4] = 9, and r[5] = 22, and the following commands are written in the run() function:
   sprn->r[4] = spro->r[2] + spro->r[3];
   sprn->r[5] = spro->r[3] - spro->r[4];
   After execution of 2 cycles, what will be the values of r2, r3, r4, and r5?

   In the end of the first cycle the values of the registers will remain the same.
   In the beginning of the second cycle the values will be: , r[2] = 12, r[3] = 45, r[4] = 57, and r[5] = 22. At the end of this cycle the values will remain the same.
   At the beginning of the 3rd cycle the values will be: , r[2] = 12, r[3] = 45, r[4] = 57, and r[5] = -12.

   1.2. The code is wrong because spro should not be modified during a clock cycle (like Q port of the FF). It should only get the sprn value in the next cycle.
2.
   2.1. The execution of the first instruction will take 7 clock cycles (Including the IDLE state). The other n-1 instructions will take 6 clock cycles each.
   So execution of n instructions will take 7+6*(n-1) = 6n+1 clock cycles.

   2.2. No, because reading from the memory can be done only when the state machine is in “EXEC0” state and the state machine does not support remaining in this state for more than 1 clock cycle.

   2.3. The disadvantage of this microarchitecture is that each instruction execution takes at least 6 clock cycles in comparison to pipelined microarchitecture.
   The advantage of this microarchitecture is that it is more simple to implement this microarchitecture than a pipelined microarchitecture.
3. The sp.c file is attached.
4. The outputs for multiplication test are under the outputs directory,
5. The outputs for Fibonacci test are under the outputs directory,
6. The sp.c file with dma is located in the with_dma directory, the outputs of the dma test are located under outputs directory and bin file under bin directory. Build with make dma.
7. ASM code:

```
    asm_cmd(ADD, 3, 0, 1,100);// 0: R3 = 100
    asm_cmd(ADD, 2, 0, 1,0);// 1: R2 = 0
    asm_cmd(ADD, 6, 0, 1,2);// 2: R6 = 2
    asm_cmd(LD, 4, 0, 3,0);// 3: R4 = MEM[R3]
    asm_cmd(ADD, 5, 0,1,1000);// 4: R5 = 1000
    asm_cmd(CMB, 0, 5, 4,10);// 5: copy the content (size 2) from address 1000 to
address that was stored in MEM[100]
    asm_cmd(POL, 6, 0, 0,0);// 6: poll the DMA status into R6
    asm_cmd(ST, 0, 6, 3,0);// 7: MEM[R3] = R6
    asm_cmd(JNE, 0, 2, 6,6);// 9: if R6 != 0 jump to pc 6
    asm_cmd(HLT, 0, 0, 0,0);// 10: HALT
```