Omar sharafi and Muhamed Biadsy Preparation report :

1) 7.1. How many pipeline stages does the SweRV EH1 processor have?

SweRV EH1 Has 9 pipeline stages.

2) 7.2. What is the maximum number of commands processed by the commit step in each cycle?

We can prosses at most 2 commands in the commit step.

3) 7.3. Provide examples of 4 different types of instruction formats as they would be decoded in the decode stage.

R-Type, I-Type, S-type(for load), UJ-type (jumps).

4) 7.4. How many cycles does it take to perform integer multiplication?

The multiply pipe contains a 3-cycle integer multiplier using three stages (M1, M2, and M3) , this is without the fetch, decode and commit (with it it will take 9 cycles)

5) 7.5. How many cycles does it take to perform division?

the processor has an out-of-pipeline divider with a 34-cycle latency , so it will take fetch, decode + out of line devide + commit -> 34 + 6 = 40 cycles.

6) 7.6. What is the difference between **Stall** and **NOP**?

A stall is a deliberate delay introduced into the processor pipeline to resolve a dependency or hazard, it is done by inserting NOPS (no-operation instructions) or using special hardware mechanisms to hold up the pipeline.

So we can say the while NOP is an instructions that does nothing, stall is using that instrction to create a deliberate delay introduced into the processor pipeline to resolve a dependency or hazard.

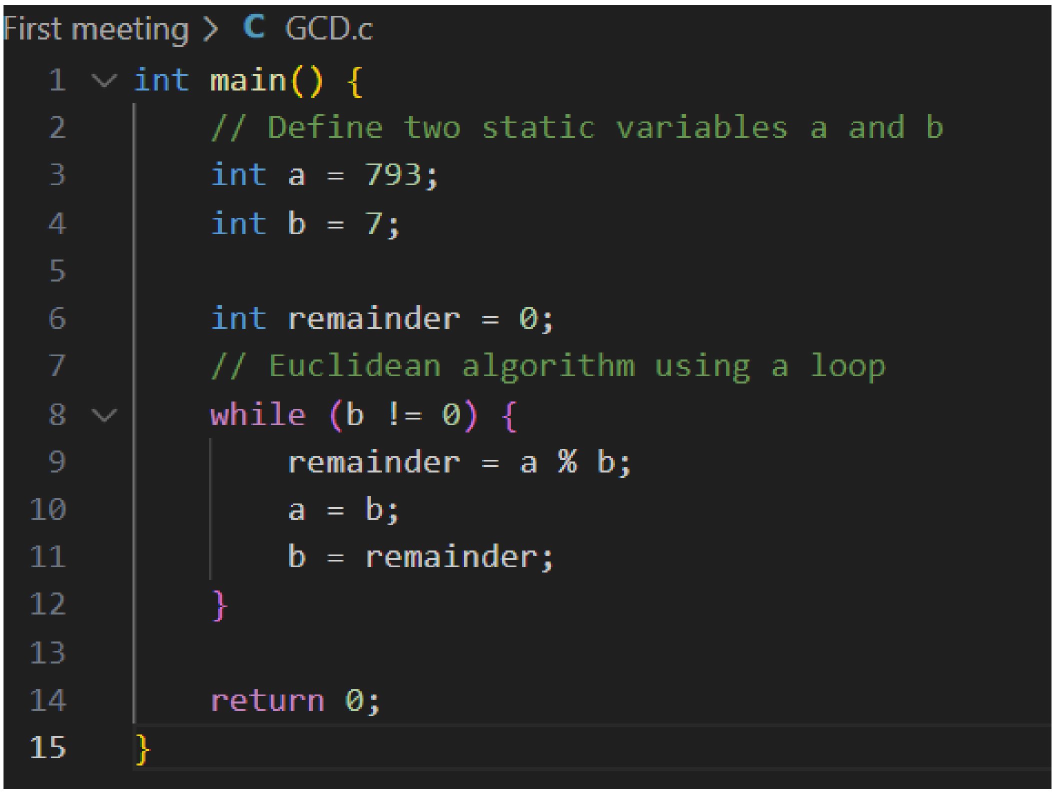
7) 7.7. How many stall points are there in the pipeline of the SweRV EH1 processor?

Four stall points (delays in the processor pipeline caused by dependencies or resource conflicts) exist in the pipeline: ‘Fetch 1’, ‘Align’, ‘Decode’, and ‘Commit’.

8) 7.8. Explain the difference between data hazards and control hazards.

A data hazard happens when you must wait for an operand to be computed from some previous step while A control hazard happens when a CPU can't tell which instructions it needs to execute next.

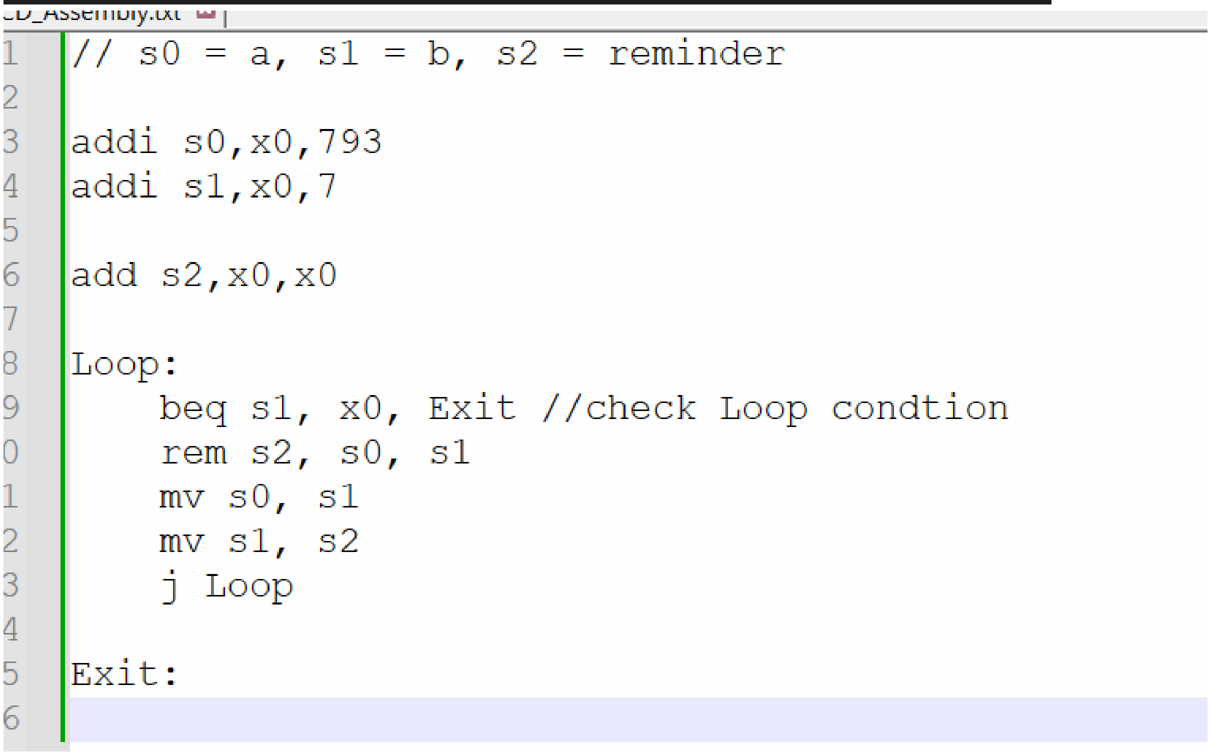
9 + 10)



```c
int main() {
    // Define two static variables a and b
    int a = 793;
    int b = 7;

    int remainder = 0;
    // Euclidean algorithm using a loop
    while (b != 0) {
        remainder = a % b;
        a = b;
        b = remainder;
    }

    return 0;
}
```



```asm
// s0 = a, s1 = b, s2 = reminder

addi s0,x0,793
addi s1,x0,7

add s2,x0,x0

Loop:
    beq s1, x0, Exit //check Loop condtion
    rem s2, s0, s1
    mv s0, s1
    mv s1, s2
    j Loop

Exit:
```