1. List and explain the different types of memory available on the RVFPGA platform.

RVfpga's Memory System has the following elements:

o External DDR Main Memory

o Cache for instructions (I$)

o Two dedicated memories, one for instructions (ICCM) and the other for data

(DCCM), which are tightly coupled to the core. These memories provide lowlatency access and SECDED ECC (single-error correction and double-error

detection error correcting codes) protection. The ICCM is disabled in the

default system.

2. How many cycles are required to access memory for each of the types of memory mentioned in the previous question?

For o External DDR Main Memory, it takes about 22 cycles.

For DCCM it takes 1 cycle.

3. In each pipeline, there are 2 ALU units, in stage EX1 and stage EX4 explain why.

In EX1, for branch instructions, the target address is calculated in this stage to enable fast branch prediction and minimize pipeline stalls.

In EX2, In the presence of a data hazard related to a multi-cycle operation, the branch is resolved at this stage.

Of course there are other reasons, like Low-latency operations: Many simple instructions can be resolved quickly in EX1 without requiring additional stages and using EX4 When facing data hazard, but we specified the reasons in contract to what the document focus on.

4. Provide two examples of code scenarios: one where the GSHARE (Global SHARE) algorithm performs well and another where it performs poorly. You don't need to write the actual code; just describe the scenarios, such as using a while loop.

A scenario where it performs well ->

```
While (x < 1000)
{
    do something....
    X++
}
```

In this case the while condition is a branch greater than assembly instruction, and since we will take the jump only one time in every 1000 iteration, the global shared mem will perform well (will take few iterations to create a history that will make us always jump, and will miss predict after that only once).

A scenario where it does not perform well ->

A instruction which jumps one time and doesn't after that
like

```
While(x<100000)
{
        If (y is even)
{
Do someting in a func1
}
Else Do someting in a func2
X++
Y++
}
```

5. In branch prediction, how many cycles of penalty are wasted if the prediction is correct? And if it is incorrect? How does the fact that there are 2 ALUs in the pipeline affect your answer?

If the branch prediction is **correct**, the penalty is **1 cycle**.

If the branch prediction is **incorrect**, the penalty depends on where the branch is resolved:

- **Primary ALU (EX1)**: 4 cycles penalty.
- **Secondary ALU (EX4)**: 7 cycles penalty.

With only the EX1 ALU, all branch resolution would need to occur in EX1. Branch instructions relying on data calculated deeper in the pipeline (e.g., multi-cycle operations) would lead delays because the required data might not yet be ready. (7 cycles)

With Having **EX4** we still need 7 cycles, but the pipe will have more instructions instead of stalls, which means having it increase Pipeline utilization .