**Lab Ex 04:**

**Question 1: Harvard architecture**

Pros:

* ROM instructions are separated from the RAM.
* Separated memory for data and instructions.
* Pipelining.
* Faster.

Cons:

* Higher implementation complexity.
* More expensive.
* Requires more space on the chip.

**Question 2: Structural and Data Hazards**

Data hazard – Usage of data that is supposed to be updated in a previous instruction led to the usage of wrong data by the program. In the current architecture, the only hazard of this kind is read after write hazard.

The solution is to check if the source registers are the destination registers of the previous instructions that weren't already updated. If so, we stall the instruction or use forwarding.

Control hazard - Happens due to the usage of branch instructions. While we have to jump to a different instruction, but the pipeline contains parts of the instruction that won’t occur due to the jump.

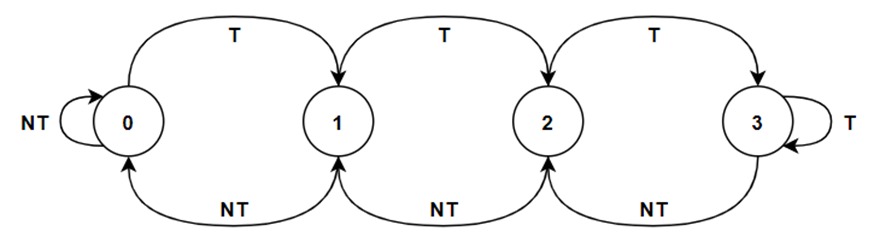
The solution is to use branch prediction.

Structural hazard - Happens when we try to read and write in the same clock cycle.

The solution is to compare the load and store registers of the instructions and stall one of the instructions according to the correct run of the program.

**Question 3: Branches and branch prediction**

We implemented a branch history table SM. The timing diagram of a single 2-bit predictor from the table:



A predictor in the table will update every jump instruction in the program according to the state machine we have drawn.

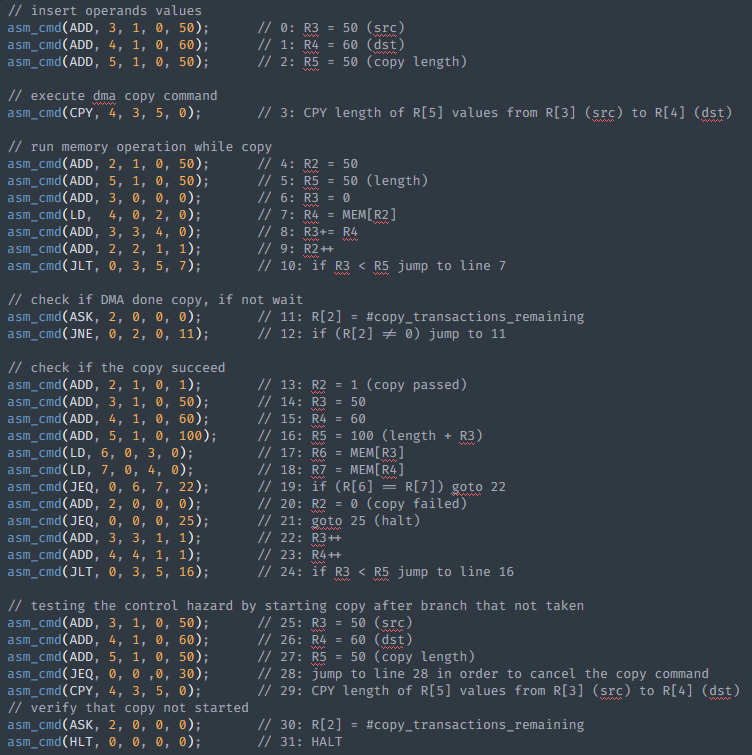
We will have 10 predictors; each predictor will be used according to the current PC. We will access the relevant predictor using operation.

**Question 6: Speedup comparison, next generation improvements**

|  |  |  |
| --- | --- | --- |
|  | **Lab 2** | **Lab 5** |
| **Fibonacci** | 1182 | 392 |
| **Multiplication** | 282 | 132 |
| **Example** | 366 | 141 |

We can see we got faster results than lab 2.  
We did not get the CPI 1 because the structural hazards we have in our problem.  
To improve our architecture we can try to avoid flushing and using better predictor. In addition, if we enlarge our current table we may get the desired results, but of course it will be more costly.

**Question 7: DMA testing in a pipeline environment**

Attach the assembly code below:

At the end of the run, we check that the CPY command will not execute after taking a branch predication.