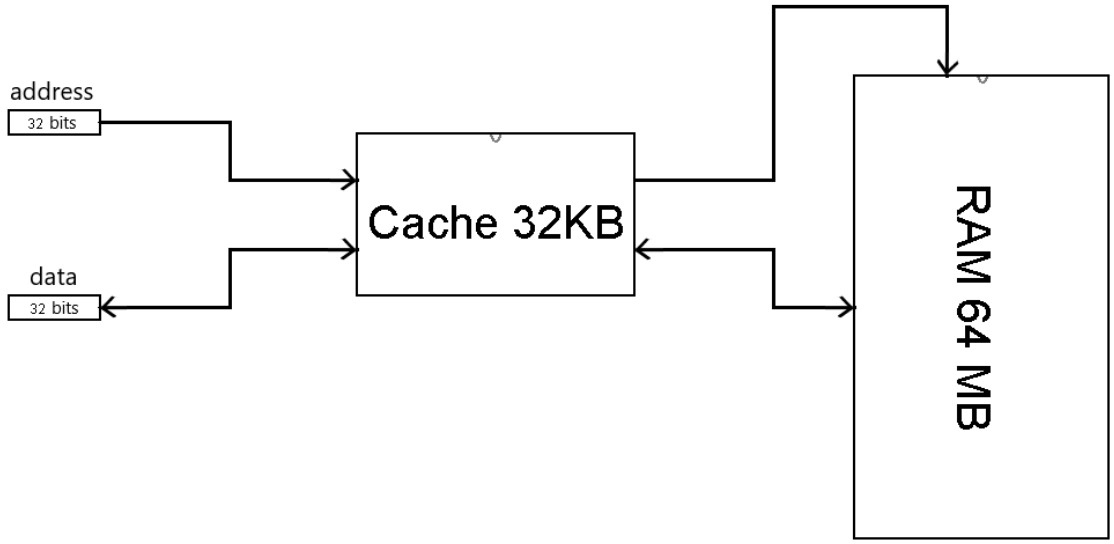
Side-project: Cache memory

1 Preamble

The project concerns the implementation of a memory cache interacting with its main memory, like depicted in the schematic below. We can see that it represents only the memory part of an 32 bits architecture without the processor. Essentially, a cache of 32 KBytes and a main memory of 64 MBytes of capacity. Normally the cache controller should be put apart from the cache memory, usually integrated to a component called the MMU (Memory Management Unit) within the CPU packaging, but in our implementation we will put them together in the same component.



We saw broadly in class how a memory cache works, but in this project we will dive to the detailed internals working of this technology, and almost all of the main parts and concepts are well described in the book "Digital Design and Computer Architecture, 2nd edition" from the author *David Harris*. You will find the essential materials in chapter 8 (*8.3. Caches* pages 480-491). This project should be implemented using Logisim. Be warned, this project is not easy.

2 Guideline

- The cache is of type 2-ways set associative, with write-back write policy and LRU (Least Recently Used) replacement policy.
- For the sake of simplicity, the cache and the main memory can only handle 32 bits words, even though the addressable cell is of 8 bits.
- Logisim can not handle the simultaneous input/output ports, which restricts the implementation to use 2 different ports, one for inputs and one for outputs.
- The response time of the cache is 1 cycle if the request hits, and many cycles otherwise. The read and write time of the main memory (RAM) is 10 cycles for one 32 bits word. You need to find the way to simulate that in Logisim.
- To overcome the synchronization problem caused by the differences in response time, we have to use a simple well known synchronization protocol. If a component wants to request a job from another, like CPU requesting data from the cache, a command line of the name of $request$ is used to inform the cache that CPU requests a data. The cache can take many cycles in the case of a "miss". Thus, a second command line is returned from the cache to the CPU called $Ready/\overline{Busy}$ informing the CPU when the cache finishes its processing and can retrieve the result. The same should be done for the cache interacting with the RAM.
- The use of external libraries is prohibited, while it is still allowed to use the internal libraries for components like, registers, counters, comparators...etc.

3 Clauses and conditions

- The circuit should be done in one Logisim file and saved like *.circ* file. The file should be sent to my email : Kara.Abdelaziz@el-kalam.com with the student information.
- The dead end for submission is for 15/01/2024 at 00:00. The solution will be exposed just after.
- Only the first five valid submissions will be accepted, the sixth and after are not awarded.
- Completing the mini-project grants you a bonus score of 15 additional points to your TP grade.
- The project is to do for one student, it is not groupe project.
- The project to be accepted needs to follow exactly the functioning terms of the guideline above.

وفقكم الله