|  |  |
| --- | --- |
| Report for Designing our own architecture.  Supervisor: DR/Hassan Soubra | by: Men in blue (team 54)   1. Hassan Tarek 43-17345 2. Emad Aboelnasr 43-16928 3. Mohamed Abdelwahab 43-18031 4. Abdelrahman Gelany 43-17100 |

* The features for our own architecture:

1. **Microarchitecture :** Von Neumann Architecture.
2. **Instruction Memory and Data Memory Size :** 1024 x 32-bits.
3. **Total Number of Registers :** 32 Registers.
4. **Instruction Format :** Instruction Set 2.

* Instruction Format :

R-type:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Opcode | Rs | Rt | Rd | Shamt |

0 5 6 10 11 15 16 20 21 31

I-type:

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Rs | Rt | Immediate |

0 5 6 10 11 15 16 31

J-type:

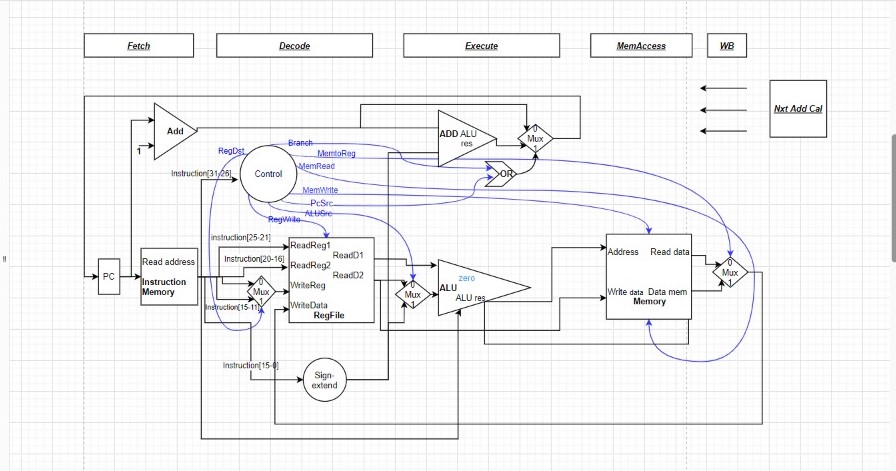
|  |  |
| --- | --- |
| Opcode | Address |

0 5 6 31

* Cache:

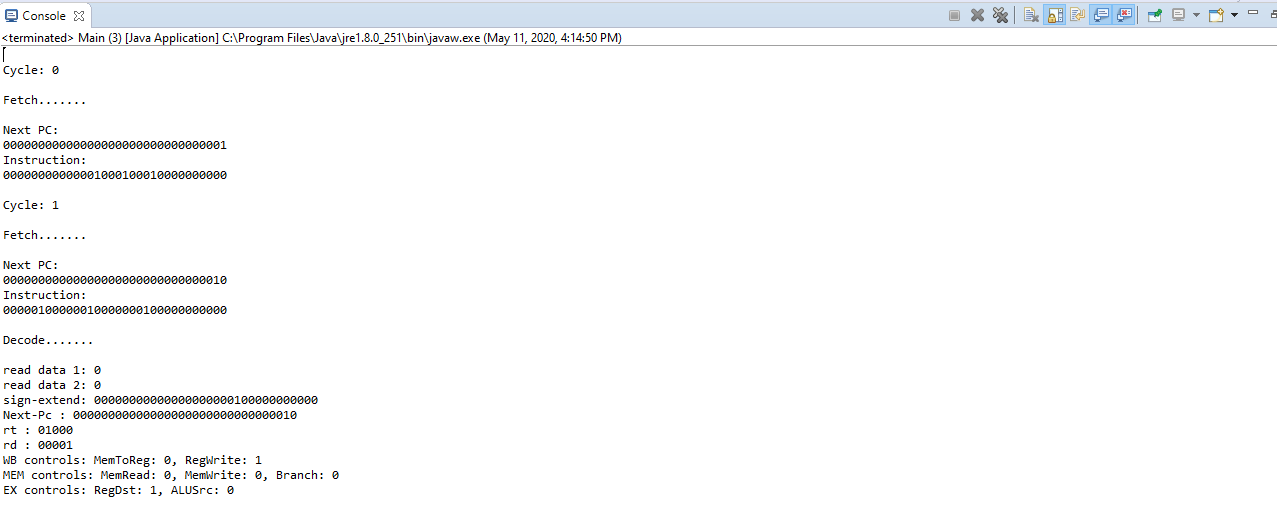
1. Type: Instruction Cache of size 100.
2. Replacement Policy: Random Replacement.

* Data Path:
* Our data path is to load instructions from the instruction cache first then the fetch stage then instruction decode the execution in the Alu then memory access and write back stage (in case of load word).



* The structure of the code :

The project is divided into two packages, one for the stages and another for the integration between the stages and experimenting the implementation. There are five stages in the architecture which are Fetch, Decode, Execute, Memory access and Write Back. In each class of the stages, there is a method that does most of the job of that stage and returns an array of objects that shall be passed to the next stage. This array of objects differs from stage to another depending on what is needed in the next stage. There exists an array in the Main Class that performs the pipelining with length of 4 representing the number of pipeline registers in the architecture. The Write Back stage does not return an array of Objects, so it does not have a pipeline register. A group of instructions is pushed into the cache and the stages begin. First by fetching an instruction, then decoding the previously fetched one and inserting the new fetched instruction in its place. After decoding an instruction, a new decoding of the older instruction starts and takes its place in the array. An execution begins at the same time of a decoding of a previous instruction then memory access an after that is the write back.

* Screenshots of the output:

