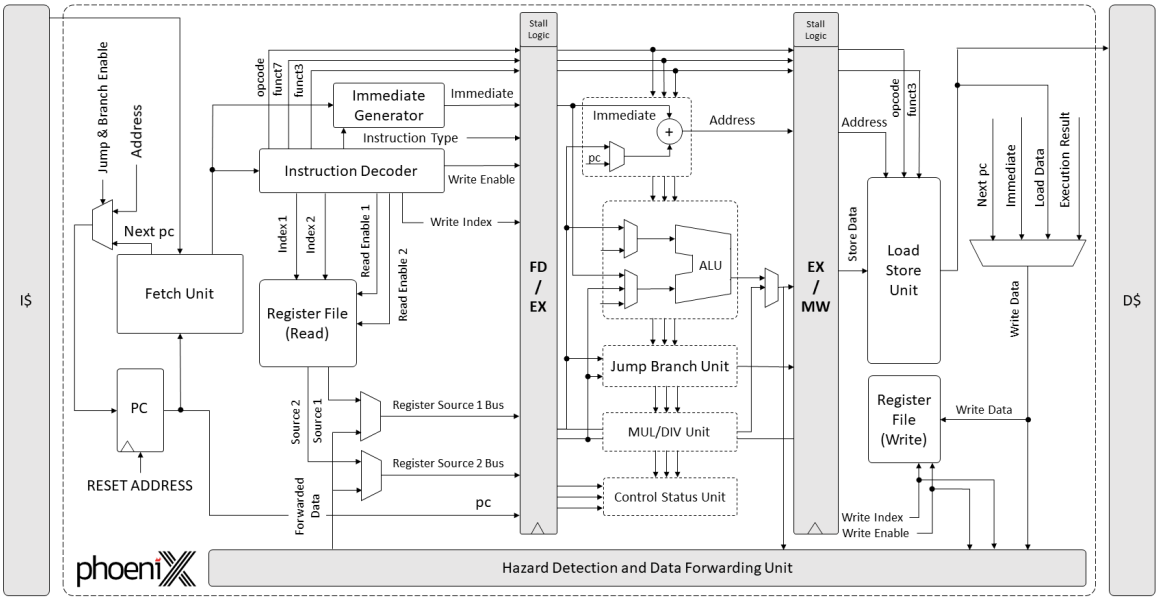
Computer Organization – Assembly Assignment

Email: iustCompOrg+4022@gamil.com

Introduction to phoeniX RISC-V Processor:

In this assignment you are going to run and execute an assembly code on **phoeniX RISC-V Processor**. The phoeniX project was initiated in the summer of 2023 at the Electronics Research Center of Iran University of Science and Technology. Led by two academic research laboratories, Super Computing and Networking (SCaN) Research Lab and the Circuit Design Research Lab. The primary objective of this project is to design a foundational processor for architectural research and development, especially in field of approximate computing.

While numerous open-source processors have been developed by universities, organizations and individuals on GitHub, many of them possess complex design and simulation processes that holds computer architecture researchers from finding a suitable platform for their research endeavors. One of the most important goals in this project was removing these complexities for developers and enthusiasts of this field, along with maintaining and increasing the quality of experiences and overall performance.



The core can be implemented as a softcore CPU on *Xilinx 6, 7, Ultrascale and Ultrascale+* series of FPGA boards using logic synthesis. This allows flexible integration of the core's functionality within the FPGA fabric.

The core has also undergone a complete synthesis flow to become an Integrated Circuit using Synopsys Design Compiler tool. The implementation was specifically carried out utilizing the *NanGate 45nm* Process Design Kit (PDK).

Here's a detailed list of phoeniX core's feature:

- Optimized 3 stage pipeline

  The 3-stage pipeline in a processor improves instruction throughput by dividing execution into sequential stages with minimal internal fragmentation. By incorporating data forwarding and bypassing options (such as forwarding data from execution, memory or writeback stage) the pipeline minimizes stalls caused by data hazards. As a result, the pipeline achieves higher performance, reduced stalls, and improved instruction-level parallelism, enabling concurrent processing of independent instructions.

- Modularity and Extensiveness

  Modularity in processor design promotes flexibility, reusability, scalability, simpler testing, and increased system reliability by breaking down the processor into smaller, independent modules that form the building blocks. Each one of these building blocks can be designed, optimized, and tested separately. This approach offers several benefits.

  First, modularity increases flexibility and reusability, as individual modules can be easily interchanged or upgraded without requiring significant changes to the main core. This enables efficient customization and adaptation to different application requirements (e.g. adding a multiplier/divider module to the design would cause significant changes to a centralized control unit, but in this methodology, designing self-controlled execution units would lead to a much simple integration of the module to the main core).

  Secondly, modularity aids in design verification and testing, as individual modules can be tested in isolated testbenches, simplifying the debugging process and reducing the overall development time.

  Additionally, modular designs can lead to improved overall system reliability, as faults and failures in one module are less likely to affect the functionality of the entire processor.

- A Novel Platform for Approximate Computing

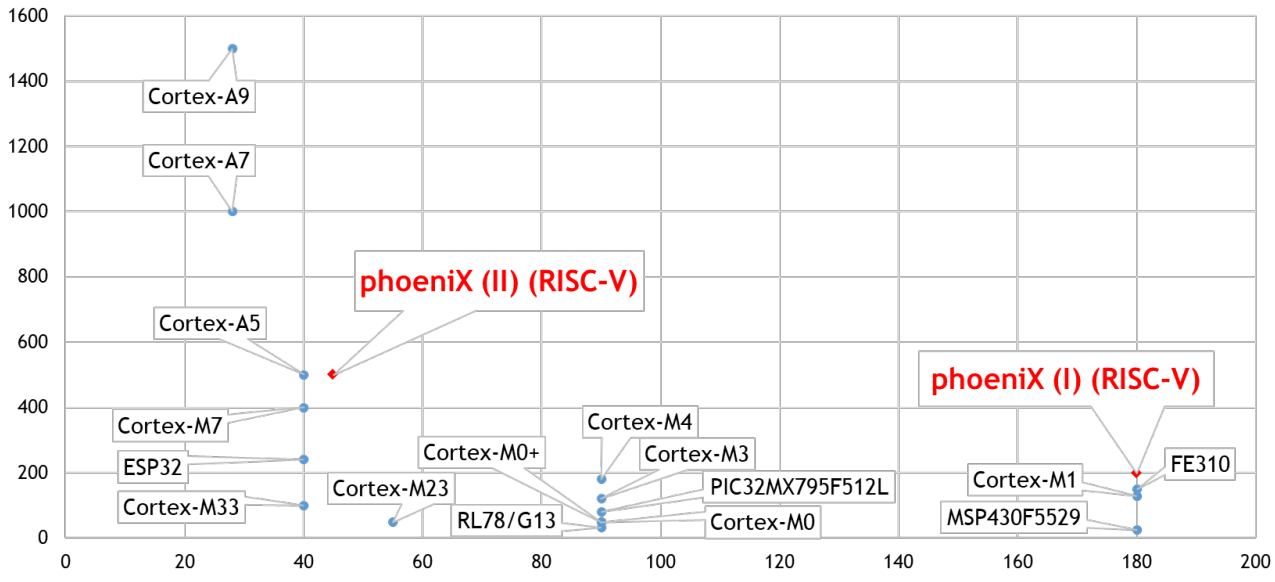
  The phoeniX RISC-V core introduces novel features that will help the emerging field of approximate computing techniques. With its modular design and extensive architecture,

phoeniX presents a configurable platform for exploring and implementing approximate computing methodologies for developers and designers.

This platform enables researchers and developers to delve into the field realm of approximate computing, where trade-offs between accuracy and computational efficiency can be carefully balanced. By offering a range of specialized instructions, optimized datapaths, and adaptable precision controls, phoeniX empowers users to use the help of approximation in diverse application domains, opening the way for advancements in energy-efficient computing, machine learning, image processing, and etc.

Processor	Max Frequency (MHz)	Technology Node (nm)	ISA	Architecture
phoeniX V0.3	620	45	RV32IEM	3-stage in order
phoeniX V0.2	500	45	RV32IM	5-stage in order
phoeniX V0.1	220	180	RV32I	5-stage in order
phoeniXS6	100 (on FPGA)	Xilinx SPARTAN6	RV32I	3-stage in order
phoeniXS7	250 (on FPGA)	Xilinx ZYNQ 7020	RV32I	3-stage in order

Here's a comparison between phoeniX core and other industrial embedded and application class processors which you may have heard of, in term of frequency:



You are going to write and execute an assembly code on phoeniX core in this assignment:

1. Star and Fork phoeniX repository to your own GitHub and clone your version of phoeniX. Here's the link of the original repository: https://github.com/phoeniX-Digital-Design/phoeniX (20 points)

```
git clone https://github.com/your_fork_of_phoeniX/phoeniX
```

2. Write, an assembly code for the given problems. Run and debug using Venus Simulator on Visual Studio Code, and then assemble the final output using the AssembleX software.

   Problem I) Write a RISC-V (RV32I) assembly code for quick sort.
   Problem II) Write a RISC-V (RV32IM) assembly code for integer square root.

   After completing this code, you'll need to run it to ensure correct execution. For this purpose, we are going to use RISC-V Venus simulator extension on VS-Code. The ISA simulator serves as our reference for the ISA. It is not cycle-accurate, but it quickly executes assembly code and provides results. Follow the instructions below:

   - Save your code in a new folder named as the same as your code file, in the *Software/User_Codes* directory.
   - In VS-Code Press Ctrl+Shift+X to open Extensions panel. Search for RISC-V Venus Simulator in the search bar and install the extension.
   - After complete installation of the extension, open your assembly code *code_name.s* and press Ctrl+F5 to run and debug it.
   - After making sure that the program runs correctly, we'll need to get an assembly text file output. From run and debug panel, choose VENUS OPTIONS go to Views and select Assembly. Save the generated file as *code_name.txt*.
   - **Important Note I**: The source code and saved text file must be in the same directory (*Software/User_Codes/Your_Code_Name/Your_Code_Name.txt*).
   - **Important Note II**: The last line in the code must be *ebreak* instruction that indicates the end of program execution, do not forget to add this line.
   - In order to create a firmware file (a file including your assembled code in hex format, in order to be mounted on processor's instruction memory) you'll need to use AssembleX software. AssembleX is a code executant assistant for the phoeniX RISC-V core, which will take *code_name* as an input, and execute the final firmware on phoeniX processor.

Waveform file will open automatically after execution, which helps you with tracing register file values to check correct execution of your code.

Enter the following command in terminal to run AssembleX software:

```
python  AssembleX_V1.0.py  code  {project_name}
```

Example:

```
python  AssembleX_V1.0.py  code  code_name
```

```
python  AssembleX_V1.0.py  code  fibonacci
```

3. Edit the README.md file according to the given sample file, and then *commit* all of your changes to the repository. The README in your GitHub repository is considered as the report of your project, so make sure to write detailed information about your code and your work. (40 points)
   - **Important Note III**: This assignment can be done in groups of 1 to 3 persons, but every team member must create the repository independently and do the mentioned tasks on their own GitHub account.

4. Download Links:
   - Visual Studio Code: https://code.visualstudio.com/download
   - GitHub Desktop: https://desktop.github.com/
   - Icarus Verilog: https://bleyer.org/icarus/
   - Python3: https://www.python.org/downloads/
   - The phoeniX project: https://github.com/phoeniX-Digital-Design/phoeniX

Please submit your homework, simulations and projects in the following format:

Name_StudentNumber_ASS1.txt (BillGates_12345678_ASS1.txt)

The text file must include the link to your GitHub repository.

Good Luck!