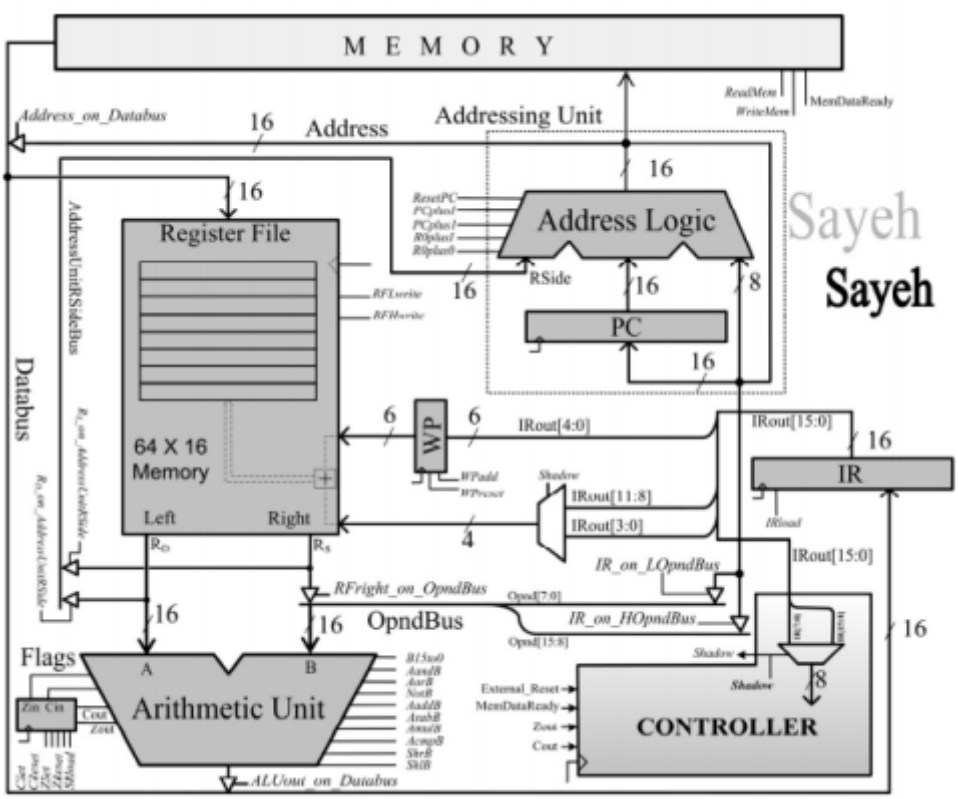
تمرین کامپیوتری شماره ۱ **تحمل‌پذیری اشکال**
نیم‌سال اول **۹۹-۹۸**
زمان تحویل: **روز سه‌شنبه ۲۸ آبان ۹۸ (ساعت ۲۳:۵۹)**

In this assignment you will get familiar with high level (RTL) fault simulation process. For the beginning you have a processor (*Sayeh*) with following architecture:



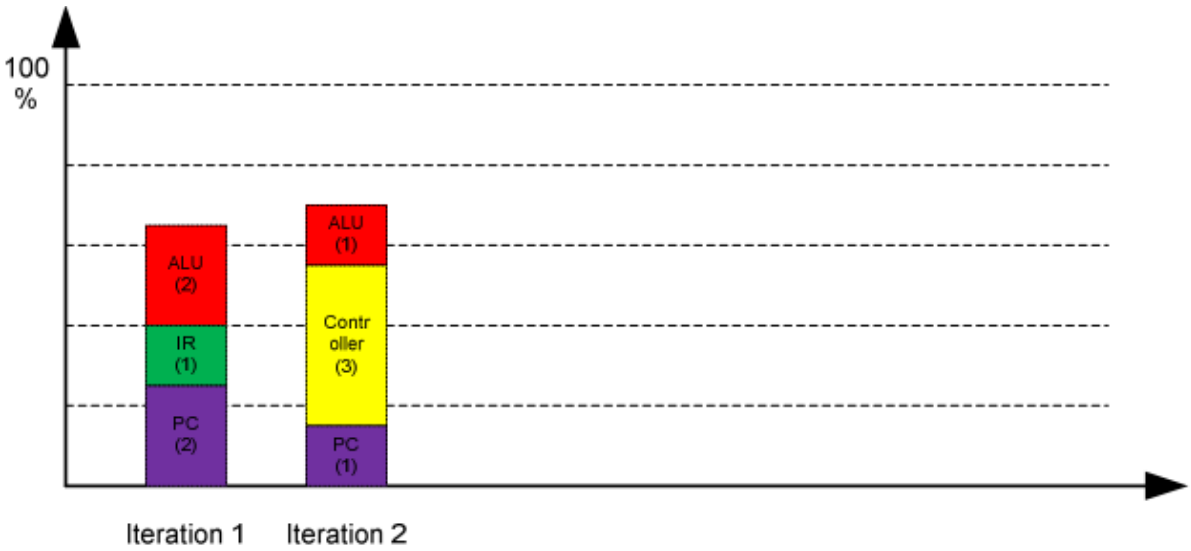
Note that there is no need to understand everything in this diagram.
For RTL fault simulation, the following steps are required in this assignment:

1. As the first step, you need to write a PLI code so that you can apply different faults into different parts of the aforementioned processor (to simplify your task, some additional documents and a video file, that explain basic concepts of PLI, are provided; In addition, a starting PLI code, a tutorial about how to configure PLI coding environment, and LRM document for PLI). Also, you can refer to “http://www.asic-world.com/verilog/pli1.html” for a simple and fast introduction about the whole concept.
2. You need to write an assembly program that adds 51 consequent numbers sequentially. After each addition, this program must write the result into the memory unit (be sure you write the result in addresses that did not conflict with the input data). You can use document of *Sayeh* processor which is uploaded with other files to learn about the instruction set of the processor. Note that this is a simple assembly loop program. As an example, assume we have 3, 5, 32, 12, and 9 as input numbers. The results which need to be written into the memory are as follows 8, 37, 44, and 21. Just make sure you select a wide range of numbers.
3. In the third phase, you need to apply 5 random (stack at) fault into the processor at the beginning of the simulation. Faults are just injected in ports and wires of different components of the processor. Then simulate the faulty processor. After each simulation of the whole program, you need to store the results, which are stored into the memory unit, to a file so you can compare it later. Note that for the first run you should simulate the processor without any injected faults and store the results of this step as your correct responses (Golden results).

4. In the next part you need to repeat the third phase 10 times. In each iteration you need to extract the following values:
    a. The location of each fault (for example in the first iteration out of 5 fault we have 2 faults in ALU, 1 fault in Controller unit, and 2 faults in Address logic)
    b. The percentage of corrupted results to the whole number of results (50)
5. For the last part, you need to apply a different number of (stuck at) fault (for example 1, 3 and 5 faults) into the different modules of the processor at the beginning of the simulation. In this part, you must determine how much the module of the processor is sensitive to the injected faults and you need to extract the percentage of corrupted results to the whole number of results (50) for each number of injected faults. Finally, you must determine which module is more sensitive to the injected faults, and explain why this result is obtained.

Report:
Your report is in a simple format but you need to provide the following parts:

- You need to upload the PLI code and its project folder + processor folder. You may be asked to simulate some parts of your results.
- Results of section 4 with a diagram that shows the percentage of the number of corrupted data in each iteration. The diagram should be like the following figure.



- Similar to section 4, the results of section 5 with a diagram that shows the percentage of the number of corrupted data for different number of injected faults in different modules.