İTÜ Computer Engineering Department
27.02.2015

Due: 11.03.2015, 23:00

COMPUTER ARCHITECTURE 1st HOMEWORK

1. A task can be done in five sequential steps. These steps take **10** ns, **15** ns, **25** ns, **30** ns, and **45** ns, respectively via dedicated hardware. The order of these steps cannot be changed. A pipeline must be designed to conduct this operation.

    a. **Discuss** how many layers should be used in this pipeline considering performance.
    b. What should be the clock period of the pipeline you have designed where the register delay is **5** ns?
    c. What is the boundary of speed up that can be obtained?
    d. Assume the task is repeated **7** times in the pipeline you have designed, when will the tasks complete? Is it possible to complete the operations earlier with another design?
    e. Assume the task is repeated **9** times in the pipeline you have designed, when will the tasks complete? Is it possible to complete the operations earlier with another design?

İTÜ Computer Engineering Department
27.02.2015

Due: 11.03.2015, 23:00

COMPUTER ARCHITECTURE 1st HOMEWORK

2. Selection Sort is a simple sorting algorithm for an array of elements. The algorithm considers an array of ***n*** elements as consisting of ***k*** sorted elements followed by ***n – k*** unsorted elements. Each step of the algorithm finds the element with the minimum value among the unsorted elements, removes it, and adds it to the end of the sorted elements. In the beginning, the value of ***k*** is zero. The algorithm terminates after each unsorted element is processed, yielding an array of ***n*** elements sorted.
   A pseudo code description of the algorithm is given below.

```
for i : firstIndex to lastIndex
       minIndex := i
       for j : i to lastIndex
              if array[j] < array[minIndex]
                     minIndex := j
       swap array[i] and array[minIndex]
```

   a. Implement the Selection Sort algorithm *elegantly* in the SPARC v8 assembly language. Your program must compile and work properly in the provided SPARC v8 simulator.
   b. List the instructions where data or branch hazards might occur in your code. Briefly explain the reason for the hazards.
   c. Fix all hazards in your code by inserting NOP instructions, where necessary.
   d. Fix all hazards in your code without affecting the performance at all, if possible. If this is not possible, explain why you cannot fix the hazards without affecting the performance.