**CS 367 - Recitation #12 Name: Simankan Kafle Architecture**

**Group Member(s): .**

**Today's Goals:** Get comfortable with logic gates, digital logic, and sequential design.

**Work in groups of 2-3 students.** Every group will turn in what they've got at the end of recitation on paper, with all the members’ names written down.

**Grading is based on participation.** Get as much done as you can. You will also be given feedback in the form of a 'score' (1-3) and possibly some comments. This doesn’t affect your grade – it is solely for feedback. A score of 3 means everything looks great.   A score of two indicates some minor problems.  And a score of one indicates that there were some major issues. If you get a 1, don't panic - go see your prof or a GTA to get more extensive feedback.

**Pipeline Efficiency**

**Task1:** In class, we discussed the idea of the speedup due to **Pipelining**. Consider:

One instruction takes 300 time units to complete, plus an overhead of 5 time units between instructions.

If we want to execute 10 instructions, how many time units are needed?

**3040**

Now, suppose the instruction is divided evenly into 5 stages, with an overhead of 5 time units between instructions. How many time units would it take to execute the same 10 instructions?

**640**

**Pipeline Hazards**

**Task2:** In class, we did a **brief** introduction into the idea of hazards in the Pipelines. A **Data Hazard** occurs when data from one instruction is needed for a future instruction, but that data hasn’t finished writing back.

Discuss the following code within your group and try and find any such hazards in each block of code.  *(hint: W has to complete before the data is in the register)*

**Refresher:** There are 5 stages for each execution (Fetch, Decode, Execute, Memory, Writeback)

**![A screenshot of a cell phone

Description automatically generated]()Code 1:**

1 **xorq %rcx, %rcx**

2 **movq $2, %rdx**

3 **movq %rdi, %rcx**

4 **addq %rdi, %rsi**

5  **orq %rdx, %rcx**

Which of the above instructions will face pipeline data hazards? *(Data not available when needed).*

For any such lines, indicate how many phases they should be delayed to execute successfully.

Lines 4 and 5 and we should delay **one** phase.

**Digital Logic**

In class, we introduced four digital logic gates: AND, OR, NOT, and XOR.

|  |  |  |  |
| --- | --- | --- | --- |
| For each gate, the inputs and output are each 1-bit values.  These gates can have any number of inputs (2, 3, or 4 inputs are all fine), but only one output. | | | |
| **Shape  Description automatically generated** | **Shape  Description automatically generated** | **Shape, arrow  Description automatically generated** |  |
| **AND** | **OR** | **NOT** | **XOR** |

For 1-bit operations, we can use the following symbols to refer to these gates in an expression:

|  |  |  |  |
| --- | --- | --- | --- |
| AND | OR | NOT | XOR |
| && | || | ! or ~ | ^ |

**Task 3:**

Write an expression for the following circuit:

|  |  |
| --- | --- |
|  | Expression:  (a && ~b) || (~(a && c)) |

**Digital Logic**

**Task 4:** In class, we introduced four digital logic gates: AND, OR, NOT, and XOR.

In this exercise, fill in the area in the dashed lines with a circuit drawing for each of the five areas to implement a very simple Arithmetic Logic Unit (ALU). You can use these four gates:

|  |  |  |  |
| --- | --- | --- | --- |
| For each gate, the inputs and output are each 1-bit values.  You can use these gates with any number of inputs (2, 3, or 4 inputs are all fine), but only one output. | | | |
|  |  |  |  |
| **AND** | **OR** | **NOT** | **XOR** |

Complete the implementation of the below ALU to perform the following functions based on s inputs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| When s0 and s1 are… | s0 = 0 and s1 = 0 | s0 = 1 and s1 = 0 | s0 = 0 and s1 = 1 | s0 = 1 and s1 = 1 |
| Perform the following… | a || b | ~ (a && b) | ~a = b | ~a ^ ~b |

a

b

s

0

s

1

a||b

~(a&&b)

~a=b

~a^~b

Out

4x1 Multiplexer