Aflevering 3 - Besvarelse

Af Peter Diekema, Sadikh Mohamed & Jesper Bertelsen

1. What is a logic gate?

* A logic gate is a component for a digital circuit. They perform logical functions by taking in specific inputs and thereby computing specific outputs. Types of logic functions could be AND, OR, NOT and so on.

2. When we build combinational circuits from logic gates, how does it “compute” the

logic it’s supposed to (e.g. how is the output generated from the input)?

* Each gate's output is computed and then fed to the next gate. In the end only one output comes out. Outputs are boolean functions of inputs. The logic comes from transistors and resistors setup in specific designs.

3. Give two advantages of the Hardware Control Language.

* Includes capability for explicit expression of time
* Do not require augmentation of extensive libraries

4. What is the HCL for “not equal”?

* “Not equal” is given by → out = !a

5. What components are Registers made of?

* A register is made up of multiple flip-flops. Each flip-flop can hold 1 bit of information. The flip-flop is made up of two latches, enabling the flip-flop to hold data between the two latches until the clock rises.

6. Is it possible to read two values from a register at the same time (in one CPU cycle)? How about reading and writing different addresses at the same time?

* When only having one register, a serial data transfer is used, and reads/writes only one value at a time.
* Reading more than one value at a time requires multiple registers, and is referred to as parallel data transfer.

7. Briefly describe the Fetch-Decode-Execute cycle.

* Fetch
  + Fetch next instruction to be executed from memory
  + PC holds the address of the next instruction to be executed
* Decode
  + Decode instruction, and send control signals to parts of datapath
  + Read register values from reg file Execute
* Execute
  + Perform specified operation on the data
  + Save results in register or memory
  + Update the PC

8. What is Stack? How do we leverage its LIFO property (e.g. what is the stack good for)?

* Stack is a storage for temporary variables created by each function
* Leveraging its LIFO property
  + Whenever a function declares a new variables, it is “pushed” onto the stack
  + Whenever a function exits, all of the variables are pushed onto the stack. This leads to the function being deleted
  + Once a stack variable is deleted, memory region becomes available for other stack variables

9. Give at least one advantage and disadvantage of the sequential processor model.

* Breaks down the operation into various logical stages
* Gives a better understanding of the execution of each machine code received

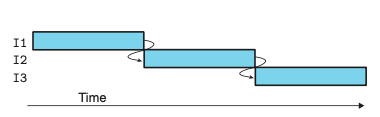
10. How does the pipelined CPU model solve the major problem of SEQ?

* Pipelined CPU model solves the major problem of SEQ with regards to the amount of instructions processed per clock cycle. This happens via

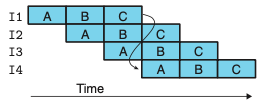
11. When is the pipelined design most effective, and why is this a design challenge?

* Pipelining is most efficient if a system has different tasks independent of each other.
* Take some data required to go through 3 tasks. When data 1 is done at the first task, then data 2 can start at task one and so forth. An unpipelined system would take data 1 through task 1-3, before data 2 would start its process, thereby speeding up the system with a pipelined system, with an ideal speedup at 3 times faster, from this example.
* The issue with a pipelined system occurs when the different processes are dependent on previous data. The so-called systems with feedback paths.   
  If data 2 were dependent on the results of data 1, but had its process started before data 1 was finished, then unexpected behavior could occur.   
  Pictures from the book illustrate the issue.

Unpipelined system:

* 

Pipelined system:

* 

12. If there are multiple stages in a pipeline that take different amounts of time to

complete, which one dominates the throughput of the whole pipeline (e.g. slowest,

fastest, closest to average, etc), and why?

* The slowest stage dominates the throughput of the whole pipeline with regards to the rate at which one can operate a clock

13. Why can’t we use an arbitrarily fast clock in a pipelined CPU?

* Because both the computational logic and registers take time to process. The register takes 20 ps, while the combinational logic can be split up depending on how many stages it is made of. Therefore the 20 ps of the register sets a theoretical limit of the clock speed.

14. Which instructions are problematic to handle in a pipelined CPU and why?

* Hazards. Instructions that are dependent on each other. This is because in a pipelined CPU multiple processes are running simultaneously. Therefore if a process depends on another this will need to be taken care of for the CPU to run properly.

15. What is a data hazard?

* A data hazard happens when dependent data modifies data in other stages of the pipeline.

16. Why is forwarding a better way to handle hazards than stalling?

* Forwarding is a better way to handle hazards than stalling for 99% of the time because it reduces the amount of cycles used to overcome the hazard, thus making it faster than stalling.

17. Give an example of a case when forwarding is not an alternative to stalling (e.g.

we must stall to ensure the correct result)?

* 100% forwarding doesn’t work on load/use data hazards. Here a combination of forwarding and stalling is required (e.g. we must stall to ensure the correct result)

18. Are you interested in designing microprocessors after this introduction of the

field?

* It seems that a microprocessor is an important part of a computer architecture without which, we will not be able to perform anything on our computers. Therefore we would be seemingly interested in designing microprocessors after this introduction of the field. This is also due to the fact that we’re in the Electrical Engineering field.