Rodrigo Becerril Ferreyra

Student ID 017584071

CECS 440 Section 02

30 November 2021

Midterm 2

Question 1

B. Instruction count is the least helpful metric.

Question 2

B. The Intel 8086 is a 16-bit processor.

Question 3

B. The product will be *n***m* bits wide.

Question 4

C. 64-bit product.

Question 5

```
 0 0 0 0 1 1 0 0 <- carry
 0 0 0 0 0 1 1 1
+0 0 0 0 0 1 1 0
 ---------------
 0 0 0 0 1 1 0 1
```

The answer is **D.**

Question 6

C. Multi-core.

Question 7

D. Anti-dependence.

Question 8

D. Parallelism.

Question 9

C. Output dependence.

Question 10

C. 4.

Question 11

C. Hazard.

Question 12

D. Both (A) and (B) (stall and bubble).

Question 13

B. 80 u.

Question 14

D. Memory access (MEM).

Question 15

A. Data hazard.

Question 16

A. IF.

Question 17

C. Assume the branch is not taken.

Question 18

A structural hazard comes up when the programmer wants to perform two or more instructions, and that combination of instructions is unsupported by the architecture. This results in undefined behavior. This happens if for example two instructions require usage of a component at the same time (perhaps two instructions need to use the ALU). This is avoided by a) not using those unsupported combinations, and b) stalling the second instruction so that it can wait its turn for using the component.

A data hazard comes up when the outcome of one instruction depends on the result of the previous one. For example, perhaps an accumulator subroutine requires the previous value of a register to be calculated before the value is incremented. Data hazards can be avoided by a) forwarding, which is passing the result of the previous instruction to the next instruction before its result is written to a register, or b) simply stalling or waiting for the previous instruction to finish (this is necessary for some instructions such as `lw`).

A control hazard comes up when the pipelined instruction is not the instruction that needed to be taken; in other words, the processor assumed that a branch was not to be taken when in fact it should have been taken. This is resolved by some kind of prediction or assumtion

(i.e. predict that the branch is not taken). If the prediction is wrong, the processor must simply flush the pipeline in order to get back to the branch it needed to take.

Question 19

The five stages are instruction fetch (IF), register read and instruction decode (ID), execution (EX), memory access (MEM), and writing the result of the instruction back to the destination register (WB). IF is the first, and it retrieves the instruction from memory. It is then decoded in ID, where the appropriate registers are read and fed into the ALU for execution (EX). If the instruction is one that modifies memory such as `lw` and `sw`, then the memory is modified or read from in the MEM stage. Lastly, the result of the entire operation is written back into the destination register in the WB stage.

Question 20

There are five addressing modes in MIPS. One of them is register addressing, which defines two source registers, a destination register, and an OP code for the ALU such as add. Another mode is immediate addressing. Instead of having two source registers, there is a field for immediate, constant values. Lastly, there is the addressing used by the jump command. This has a very large immediate field which is then added to the program counter (PC) to calculate a jump to another instruction.

Question 21

Pipelining is a technique used to speed up processing of data. It works by separating the tasks of an instruction so that multiple instructions can run simultaneously instead of one instruction having to wait on another. For example, while one instruction is calculating a sum,

the next instruction can be getting ready to calculate a difference. Pipelining is very important in processor design.

Question 22

Data hazards occur when the outcome of the current instruction relies on the outcome of the previous instruction. There are two solutions to this problem: first, the current instruction can wait for the previous to complete before starting. This is called stalling, and is necessary for certain combinations of instructions. However, this approach is very slow; it defeats the purpose of pipelining since instructions must execute one at a time. Another solution to data hazards is called forwarding. This works by taking the result of the previous instruction before it is written into a register and giving it to the current instruction to be calculated. This allows for pipelining to work as intended and makes sure that the instructions don't need to wait on one another.

Question 23

	Clock Cycle													
Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14
LOAD R1, #12(R3)	IF	ID	EX	MEM	WB									
NOP		STALL	STALL	STALL	STALL	STALL								
NOP			STALL	STALL	STALL	STALL	STALL							
ADD R6, R1, R7				IF	ID	EX	MEM	WB						
NOP					STALL	STALL	STALL	STALL						