Computer Architecture

BLG 322E

Homework 1 Report

KAAN KARATAŞ

karatask20@itu.edu.tr

Faculty of Computer and Informatics Engineering

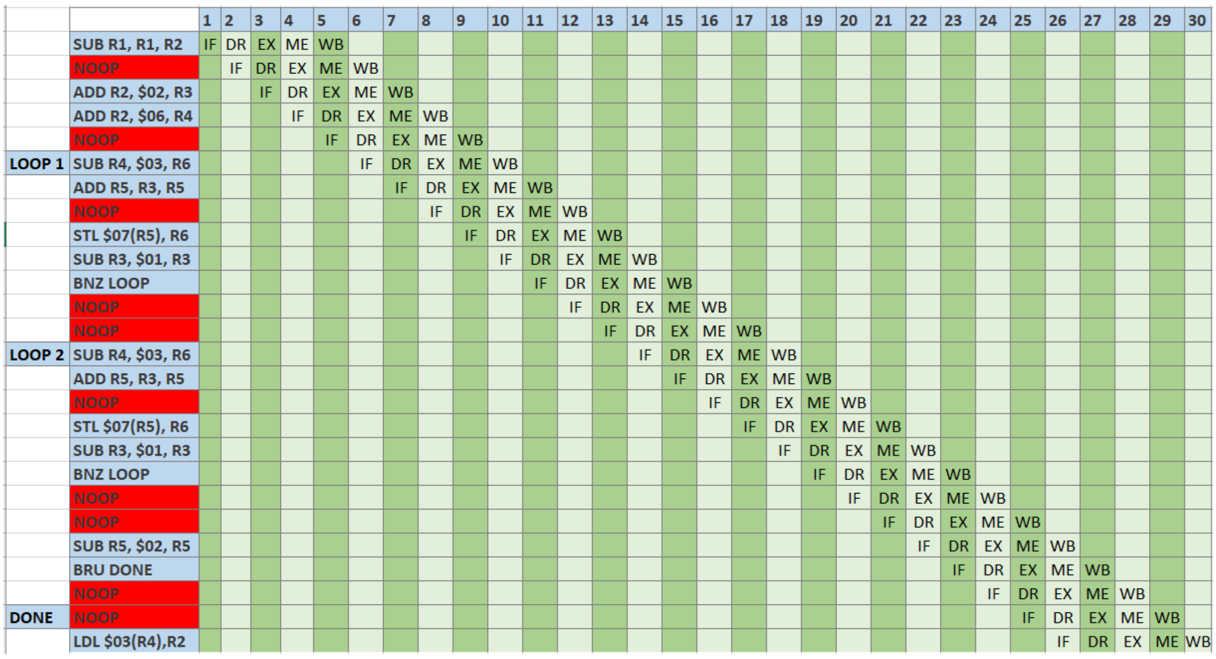
Department of Computer Engineering

Date of submission: 27.03.2023

1. Question 1

1.1. A - Time Diagram

The space-time diagram for the execution of the program wanted in question A is given below:



		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	SUB R1, R1, R2	IF	DR	EX	ME	WB																									
	NOOP		IF	DR	EX	ME	WB																								
	ADD R2, $02, R3			IF	DR	EX	ME	WB																							
	ADD R2, $06, R4				IF	DR	EX	ME	WB																						
	NOOP					IF	DR	EX	ME	WB																					
LOOP 1	SUB R4, $03, R6						IF	DR	EX	ME	WB																				
	ADD R5, R3, R5							IF	DR	EX	ME	WB																			
	NOOP								IF	DR	EX	ME	WB																		
	STL $07(R5), R6									IF	DR	EX	ME	WB																	
	SUB R3, $01, R3										IF	DR	EX	ME	WB																
	BNZ LOOP											IF	DR	EX	ME	WB															
	NOOP												IF	DR	EX	ME	WB														
	NOOP													IF	DR	EX	ME	WB													
LOOP 2	SUB R4, $03, R6														IF	DR	EX	ME	WB												
	ADD R5, R3, R5															IF	DR	EX	ME	WB											
	NOOP																IF	DR	EX	ME	WB										
	STL $07(R5), R6																	IF	DR	EX	ME	WB									
	SUB R3, $01, R3																		IF	DR	EX	ME	WB								
	BNZ LOOP																			IF	DR	EX	ME	WB							
	NOOP																				IF	DR	EX	ME	WB						
	NOOP																					IF	DR	EX	ME	WB					
	SUB R5, $02, R5																						IF	DR	EX	ME	WB				
	BRU DONE																							IF	DR	EX	ME	WB			
	NOOP																								IF	DR	EX	ME	WB		
DONE	NOOP																									IF	DR	EX	ME	WB	
	LDL $03(R4),R2																										IF	DR	EX	ME	WB

Figure 1.1: Space-Time Diagram Created With NOOP

In constructing the space-time diagram, both both data and branch hazards are addressed, single NOOP instructions are inserted to halt the pipeline as of the bypass connection between the ALU inputs and the ME stage, stalling the pipeline when necessary. Furthermore, to circumvent branch hazards, two NOOP instructions are introduced, placed to accommodate the two-cycle delay resulted by the computation of both branching addresses and flags within the EX stage.

The total penalty obtained due to these operations amounts to 10 clock cycles, each NOOP instruction serving as one penalty.

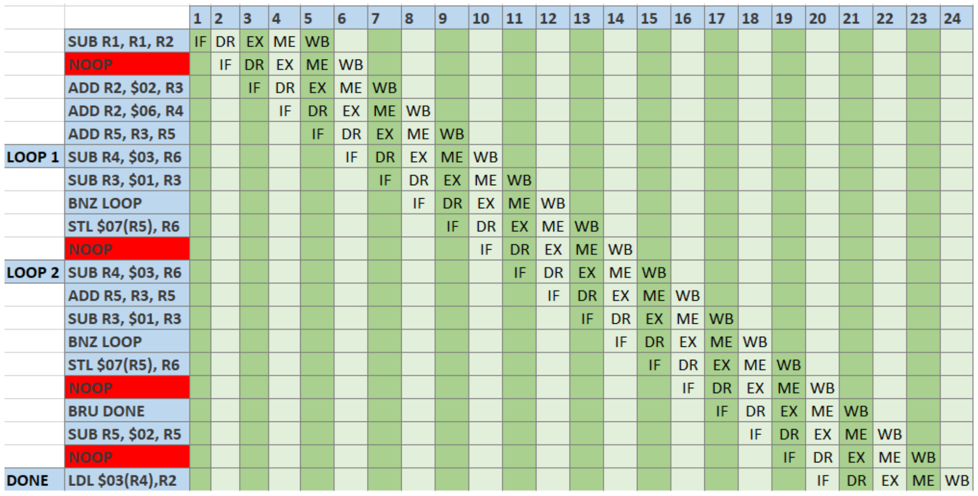
1.2. B - Total Penalty in N

The total penalty resulting from branch and data risks can be computed as 3n + 4 in the case when the loop iteration count is denoted by ’n’. This calculation is based on the inclusion of a constant of four NOOP instructions outside the loop and three NOOP instructions inside the loop that are repeated ’n’ times. This concise formula provides a clear picture of the total influence on program execution efficiency by encapsulating the amount of delay caused by hazard reduction measures.

1.3. C - Total Penalty in N

The cycle per instruction (CPI) is calculated as the total number of cycles divided by the total number of instructions, yielding 30/26 $\approx$ 1.15 cycles per instruction.

1.4. D - Optimized Pipeline



		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	SUB R1, R1, R2	IF	DR	EX	ME	WB																			
	NOOP		IF	DR	EX	ME	WB																		
	ADD R2, $02, R3			IF	DR	EX	ME	WB																	
	ADD R2, $06, R4				IF	DR	EX	ME	WB																
	ADD R5, R3, R5					IF	DR	EX	ME	WB															
LOOP 1	SUB R4, $03, R6						IF	DR	EX	ME	WB														
	SUB R3, $01, R3							IF	DR	EX	ME	WB													
	BNZ LOOP								IF	DR	EX	ME	WB												
	STL $07(R5), R6									IF	DR	EX	ME	WB											
	NOOP										IF	DR	EX	ME	WB										
LOOP 2	SUB R4, $03, R6											IF	DR	EX	ME	WB									
	ADD R5, R3, R5												IF	DR	EX	ME	WB								
	SUB R3, $01, R3													IF	DR	EX	ME	WB							
	BNZ LOOP														IF	DR	EX	ME	WB						
	STL $07(R5), R6															IF	DR	EX	ME	WB					
	NOOP																IF	DR	EX	ME	WB				
	BRU DONE																	IF	DR	EX	ME	WB			
	SUB R5, $02, R5																		IF	DR	EX	ME	WB		
	NOOP																			IF	DR	EX	ME	WB	
DONE	LDL $03(R4),R2																				IF	DR	EX	ME	WB

Figure 1.2: Space-Time Diagram Of Optimized Pipeline

The strategic use of NOOP instructions helps resolve conflicts in the instruction pipeline, enhancing CPU efficiency by reallocating instructions. By relocating certain operations, such as ADD and SUB instructions to higher levels and STL and SUB instructions to lower levels, conflicts are minimized. Specifically, I replaced suitable NOOP instructions with appropriate operations.

This restructuring allows for the execution of 20 instructions that include 4 NOPs within 24 clock cycles, highlighting the impact of strategic instruction placement on CPU efficiency.

2. Question 2

2.1. A - One-bit, Take Branch

Examining the branch prediction pattern identifies a unique set of results. The bit is initially left unaltered with the prediction bit set to 1 and the branch chosen in the first iteration since mod 3 of 30 is equal to zero. But the branch's non-execution causes a misprediction in the next iteration with a counter value of 29, which leads to the bit being set to 0. In the subsequent iteration, when the counter is 28, this modification turns out to be accurate; nevertheless, it causes a misprediction for counter 27, which resets the prediction bit to 1. This pattern continues, generating a sequence of "correct, incorrect, incorrect..." after every third iteration beginning at number 28.

When it comes to "1-bit dynamic prediction," the cumulative result shows 19 incorrect predictions and 11 right predictions if the starting prediction is to take the branch.

2.2. B - Two-bits, Take Branch

Examining the branch prediction pattern, we observe an initial correct prediction succeeded by mispredictions for counters 29 and 28, followed by a repeating sequence of "incorrect-correct-correct..." This consistent pattern results in 11 mispredictions and 19 correct predictions overall.

2.3. C - Two-bits, Not Take Branch

Upon further examination of the branch prediction pattern, we note an initial prediction of "not to take the branch (00)" in the first iteration, which correctly results in the branch being taken due to mod 3 of 30 being zero. Subsequent adjustments to the prediction bits occur, with correct predictions occurring for counters 29 and 28. However, counter 27 results in a misprediction, resetting the prediction bits. This sequence of "incorrect, correct, correct..." repeats starting from 27, resulting in 20 correct predictions and 10 false predictions overall.