CS3350 Assignment 4
Mazen Baioumy
250924925

Answer To Question 1:

IF/ID → What goes in is the current instruction encoding and the address of next one. In this case both are 32 bits, so it will be (32)(2) = 64 bits

ID/EX → We get 5 bits from the two instructions at [20-16], and [15-11], we also get 32 bits from the sign-extender instruction [15-0]. We get two 32 bits from reg1 and reg 2 in the data read through register, which adds up to 64 from (read data 1 and 2). We get 32 bits from address of instruction which is 32 bits. We also get 2-bit, 3-bit and 5-bit from the control signals for WB, M and EX. Therefore, it all adds up to →
5+5+32+32+32+32+3+2+5 = 148 bits.

EX/MEM → We get 2 bits from WB, and 3 bits from M from the control signals. We also get 32 bits from ALU result. We get 32 bits from the result from the adder, 1 from the zero-flag bit, 32 bits from (Read data 2) result, and 5 bits from the mux. Therefore, it adds to → 2+3+32+32+1+32+5 = 107 bits.

MEM/WB → We get 2 bits from WB control signals, 32 bits from Read data, 32 bits from ALU result, and 5 bits from mux. Therefore, it all adds up to → 32+32+2+5 = 71 bits.

Answer To Question 2:

Part A) To compute the actual pipeline speedup, we first need to find the total execution time for both the non-pipelined and pipelined cases. Then, we can take the ratio of the two.
Non-pipelined execution time: Each instruction takes N-1 stages with t units of time and one stage with m*t units of time. Therefore, the total non-pipelined execution time for c instructions is:
non-pipelined = c * ((N - 1) * t + m * t)
Pipelined execution time: In a pipelined processor, the first instruction takes N-1 stages with t units of time and one stage with mt units of time. Subsequent instructions can be processed in parallel, with each instruction taking an additional t unit of time for the first N-1 stages, and then mt units of time for the last stage. So, the total pipelined execution time is:
pipelined = [(N - 1) * mt] + ((c) * mt)
Now, we can compute the actual pipeline speedup as the ratio of non-pipelined to pipelined execution time:

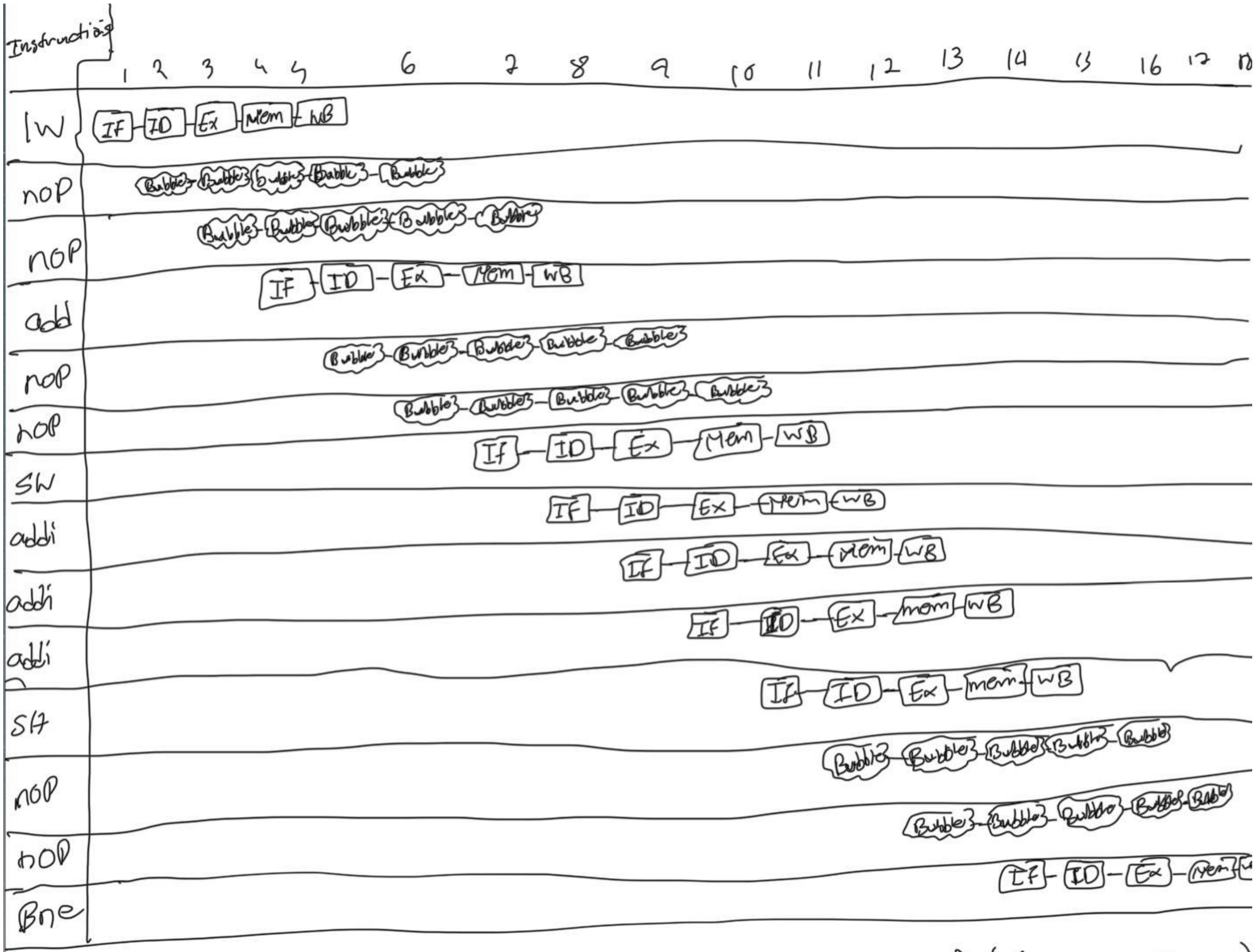
$$\text{Speedup} = \text{non-pipelined} / \text{pipelined} = \frac{c * [((N - 1) * t) + (m * t))]}{[(N - 1) * mt] + (c * mt)}$$

Part B) Let's analyze the pipelined processor and compute the expression for the ratio of time for which the pipeline runs at full occupancy.
Pipeline filling time (initial phase): In this phase, the first stage of the pipeline starts processing the first instruction. The pipeline is being filled with instructions.
The time for the pipeline to reach full occupancy is given by: Time to fill = (N - 1) * t
Pipeline processing time (steady phase): Once the pipeline is full, it enters the steady phase. During this time, the pipeline processes one instruction per t units of time for the first (N - 1) stages and m * t units of time for the last stage. We have a total of c instructions. Since the last stage is slower than the others, it takes longer for an instruction to finish. Thus, the time it takes to process c instructions will be dominated by the last stage. Therefore, time processing = c * m * t
Pipeline emptying time (final phase): After the last instruction has entered the pipeline, the pipeline starts to empty. The time it takes for the pipeline to become empty depends on the first (N - 1) stages. Time to empty = (N - 1) * t
Now, we can compute the total time the pipeline takes to process all the instructions:
Total time = Time to fill + Time processing + Time to empty Total time = (N - 1) * t + c * m * t + (N - 1) * t Total time = 2 * (N - 1) * t + c * m * t
The time for which the pipeline runs at full occupancy is given by the time spent in the processing phase, when all stages are actively computing work:
Full occupancy time = Time processing = c * m * t
The ratio of time for which the pipeline runs at full occupancy:
Full occupancy ratio = Full occupancy time / Total time Full occupancy ratio =

$$\frac{(c * m * t)}{(2 * (N - 1) * t + c * m * t)}$$

Answer To Question 3:

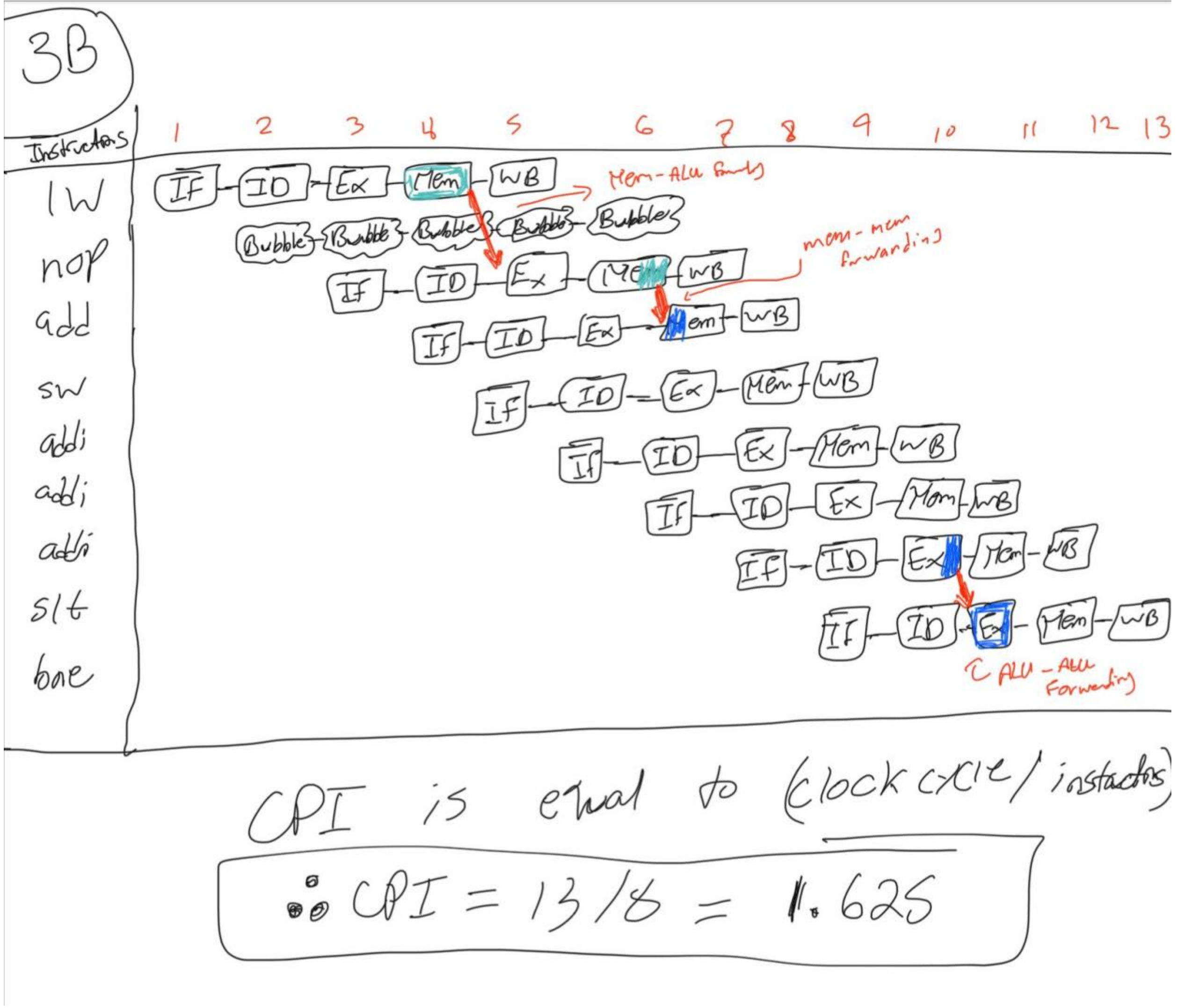
Part A.



∴ CPI is equal to → (clock cycles) / (number of instructions)

$$\boxed{CPI = 18/8 = 9/4 = 2.25}$$

Part B.



CPI is equal to (clock cycle / instructions)

$\therefore$ CPI = 13/8 = 1.625

Part C.

```
        add $t0, $0, $0
loop: lw $s1, 0($s4)
        lw $s3, 4($s4)
        add $s0, $s1, $t0
        addi $t0, $t0, 1
        add $s6, $s3, $t0
        sw $s0, 0($s2)
        sw $s6, 4($s2)
        addi $t0, $t0, 1
        addi $s2, $s2, 8
        addi $s4, $s4, 8
        slt $t2, $t0, $s5
        bne $t2, $0, loop
```

Part D.

	ALU or Branch	Data Transfer	CC
loop:	Nop	lw $s1,0($s4)	1
	add $0, $s1, $t0	lw $s3, 4($s4)	2
	addi $t0, $t0, 1	nop	3
	add $s6, $s3, $t0	sw $s0, 0($s2)	4
	addi $t0, $t0, 1	sw $s6, 4($s2)	5
	addi $s2, $s2, 8	nop	6
	addi $s4, $s4, 8	nop	7
	slt $t2, $t0, $s5	nop	8
	Bne $t2, $0, loop	nop	9

Answer to Question 4:

Time	Memory Access	Hit/Miss Type	Time	Memory Access	Hit/Miss Type
1	P1 Reads 5	Cold Miss	11	P2 Writes 9	Hit
2	P2 Writes 8	Cold Miss	12	P2 Writes 10	Cold Miss
3	P1 Reads 9	Cold Miss	13	P2 Reads 2	Cold Miss
4	P1 Writes 14	Cold Miss	14	P1 Writes 7	Cold Miss
5	P1 Reads 3	Cold Miss	15	P1 Reads 8	False share
6	P1 Writes 12	Cold Miss	16	P1 Reads 4	Capacity Miss
7	P2 Reads 6	Cold Miss	17	P2 Reads 12	Cold Miss
8	P2 Reads 17	Cold Miss	18	P2 Reads 7	True Share
9	P1 Reads 20	Cold Miss	19	P1 Writes 2	Hit
10	P2 Reads 4	Cold Miss	20	P1 Reads 11	Cold Miss

P1 Cache

Set	Cache Block Data		State
0	8	9	I
1	2	3	M
	10	11	S
2	20	21	E
	12	13	S
3	14	15	M
	6	7	S

P2 Cache

Set	Cache Block Data		State
0	8	9	S
	16	17	E

1	10	11	S
	2	3	I
2	4	5	E
	12	13	S
3	6	7	S