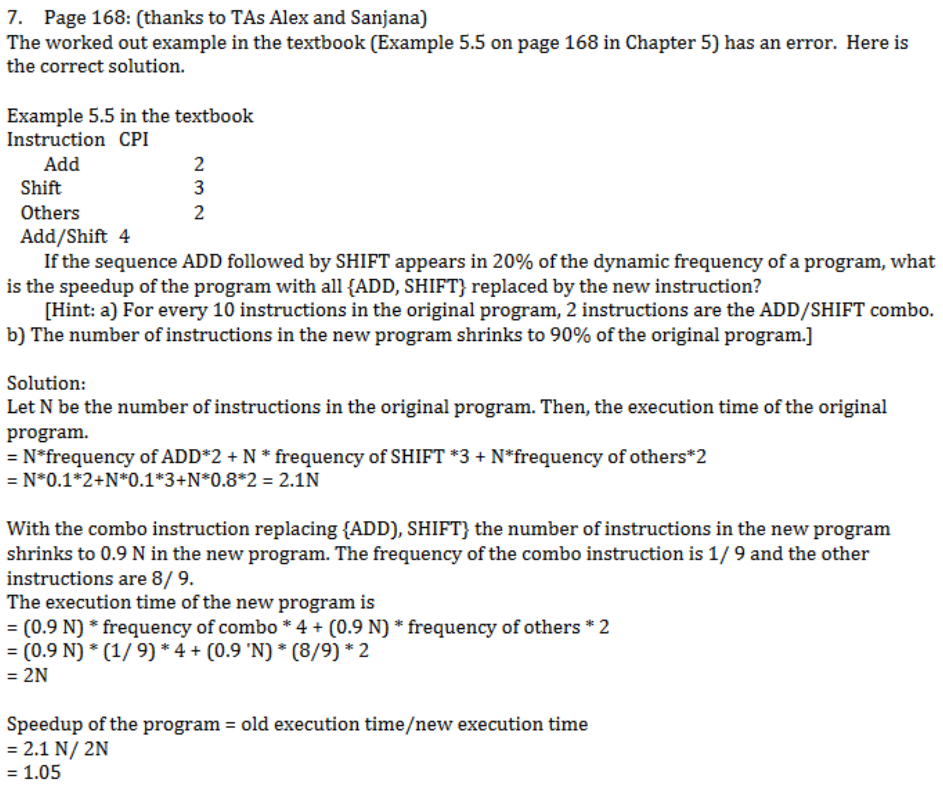


7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

Exam 2

Maybe we can put our answers in red?
Please set the google docs mode to suggesting. Not editing. JUST UPDATED PERMISSIONS

Question 1. Pipeline Buffer (11 points) (6 minutes)

List the minimum size and contents of each pipeline buffer needed to execute the SW instruction of the LC-2200 ISA. Remember that this is a **32-bit** machine. If a certain field is not stored in a given buffer, leave that table entry empty.

Commented [1]: I don't think we need Rx, since once we decode it, we don't need to know which register it came from for the rest of the instruction. We would if it was a load word, since we'd need to know where to store it, but not for storing a word in memory

Commented [2]: agree

Commented [3]: Do we need anything other than OPCode in MBUFF? Since we don't actually do anything related to register writing, do we need anything other than OPCode, which tells the pipeline not to do anything?

7. Page 168: (thanks to TAs Alex and Sanjana)

The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:

Let N be the number of instructions in the original program. Then, the execution time of the original program.

= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2

= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.

The execution time of the new program is

= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2

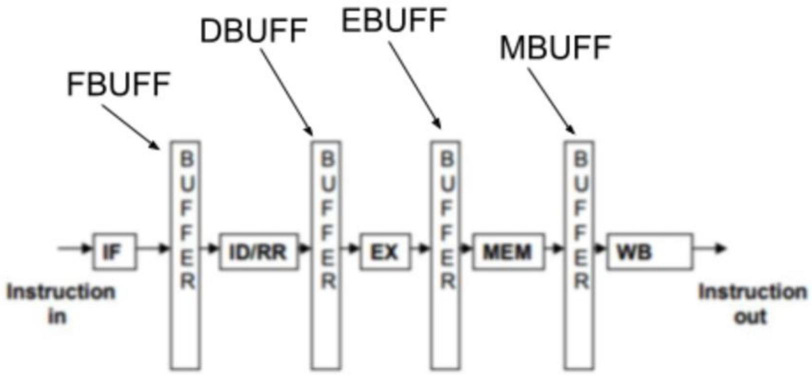
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2

= 2N

Speedup of the program = old execution time/new execution time

= 2.1 N/ 2N

= 1.05



Reference: SW Rx, Offset(Ry) SW: mem[value of Ry + offset] = value of Rx

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

Field	FBUFF	DBUFF	EBUFF	MBUFF
Instruction	32			
OPcode		4	4	4
Rx				
Ry				
Rz				
Decoded Rx		32	32	
Decoded Ry		32		
Decoded Rz				

Commented [4]: I thought we are writing the data in ...

Commented [5]: No. - From the textbook: "for SW ...

Commented [6]: Can you point to me which part of t ...

Commented [7]: ok i found it but why none of the ...

Commented [8]: The MEM stage requires the D-ME ...

Commented [9]: However, why is it just the opcode? ...

Commented [10]: The output is there to tell the ...

Commented [11]: The opcode*

Commented [12]: I believe writing to the mem is dor ...

Commented [13]: ok, the mem STAGE (not buffer) ...

Commented [14]: unfortunately, textbook isn't very ...

Commented [15]: The write back stage writes back t ...

Commented [16]: pretty sure nothing goes in MBUF ...

Commented [17]: According to 5-26 in the textbook, ...

Commented [18]: ^

Commented [19]: The MEM stage requires the D- ...

Commented [20]: From the textbook: "for SW ...

Commented [21]: I think it should be nothing, they're ...

Commented [22]: We must include opcode to say w ...

Commented [23]: _Marked as resolved_

Commented [24]: _Re-opened_

Commented [25]: Why is this 4 and not 32

Commented [26]: we need to store the value at Rx ...

Commented [27]: why would u need the rx specifier

Commented [28]: Rx is still needed because you ha ...

Commented [29]: I feel like this has been switched ...

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

Field	FBUFF	DBUFF	EBUFF	MBUFF
SEXT Offset (SEXT MEANS SIGN EXTENSION!!!!! NOT 20 EVER!!!!!!!!!!!)		32		
ALU Output (Ry + Offset)			32	
MEM[address]				
TOTAL Size:	32	100~~72~~	68~~40~~	4

Commented [4]: I thought we are writing the data in

Commented [5]: No. - From the textbook: "for SW

Commented [6]: Can you point to me which part of t

Commented [7]: ok i found it but why none of the

Commented [8]: The MEM stage requires the D-ME

Commented [9]: However, why is it just the opcode?

Commented [10]: The output is there to tell the

Commented [11]: The opcode*

Commented [12]: I believe writing to the mem is dor

Commented [13]: ok, the mem STAGE (not buffer)

Commented [14]: unfortunately, textbook isn't very

Commented [15]: The write back stage writes back t

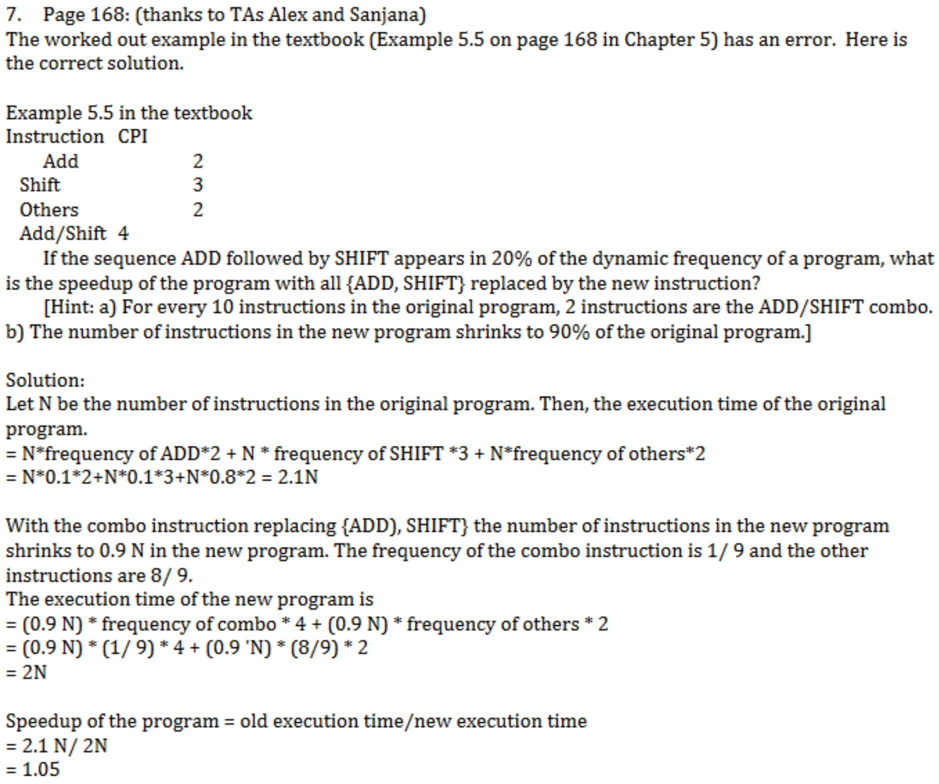
Commented [30]: when do we allocate space for this?

Commented [31]: would it be in EBUFF since we on

Commented [32]: We do not need to allocate the da

Commented [33]: So this is just a redundant row to

Commented [34]: i think you use this for LW



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

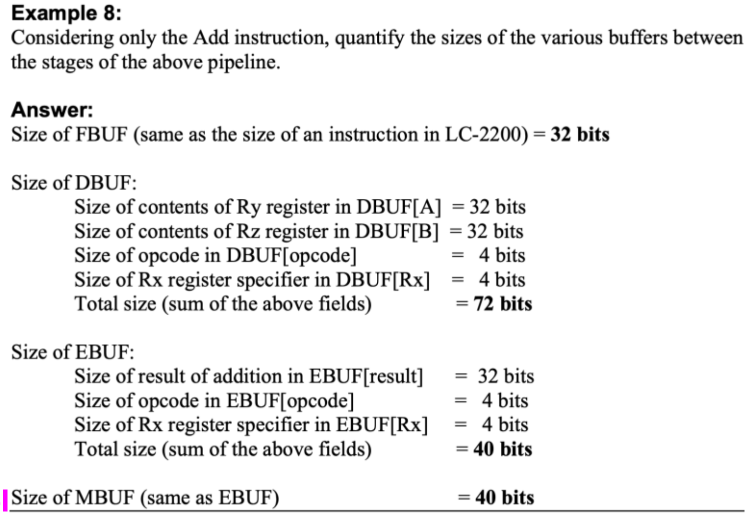
[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

Textbook example 5.8



Example 8:
Considering only the Add instruction, quantify the sizes of the various buffers between the stages of the above pipeline.

Answer:
Size of FBUF (same as the size of an instruction in LC-2200) = **32 bits**

Size of DBUF:

Size of contents of Ry register in DBUF[A]	= 32 bits
Size of contents of Rz register in DBUF[B]	= 32 bits
Size of opcode in DBUF[opcode]	= 4 bits
Size of Rx register specifier in DBUF[Rx]	= 4 bits
Total size (sum of the above fields)	= **72 bits**

Size of EBUF:

Size of result of addition in EBUF[result]	= 32 bits
Size of opcode in EBUF[opcode]	= 4 bits
Size of Rx register specifier in EBUF[Rx]	= 4 bits
Total size (sum of the above fields)	= **40 bits**

Ppl

Size of MBUF (same as EBUF) = **40 bits**

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

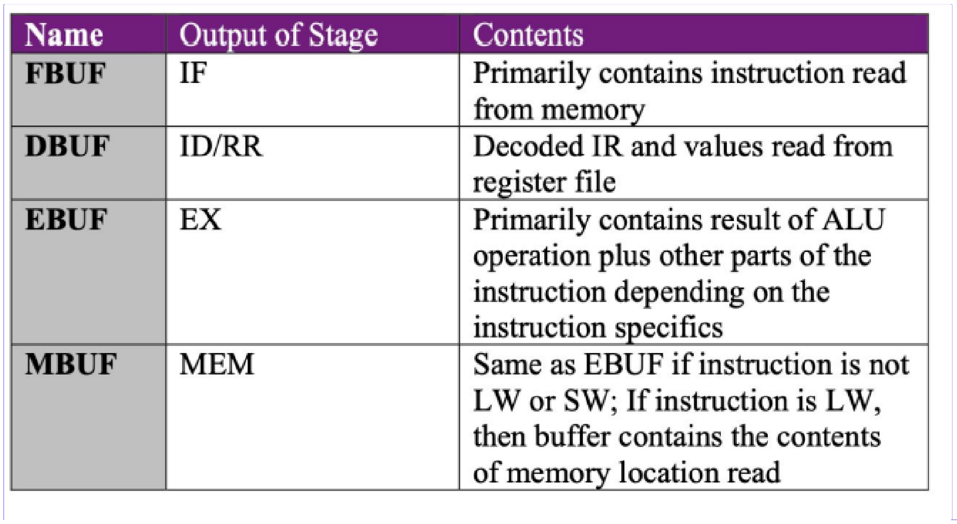
[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

According to table below MBUF should only contain opcode



Name	Output of Stage	Contents
FBUF	IF	Primarily contains instruction read from memory
DBUF	ID/RR	Decoded IR and values read from register file
EBUF	EX	Primarily contains result of ALU operation plus other parts of the instruction depending on the instruction specifics
MBUF	MEM	Same as EBUF if instruction is not LW or SW; If instruction is LW, then buffer contains the contents of memory location read

Commented [35]: the textbook says we copy MBUF from EBUF when the instruction is not LW or SW. But what about LW and SW?

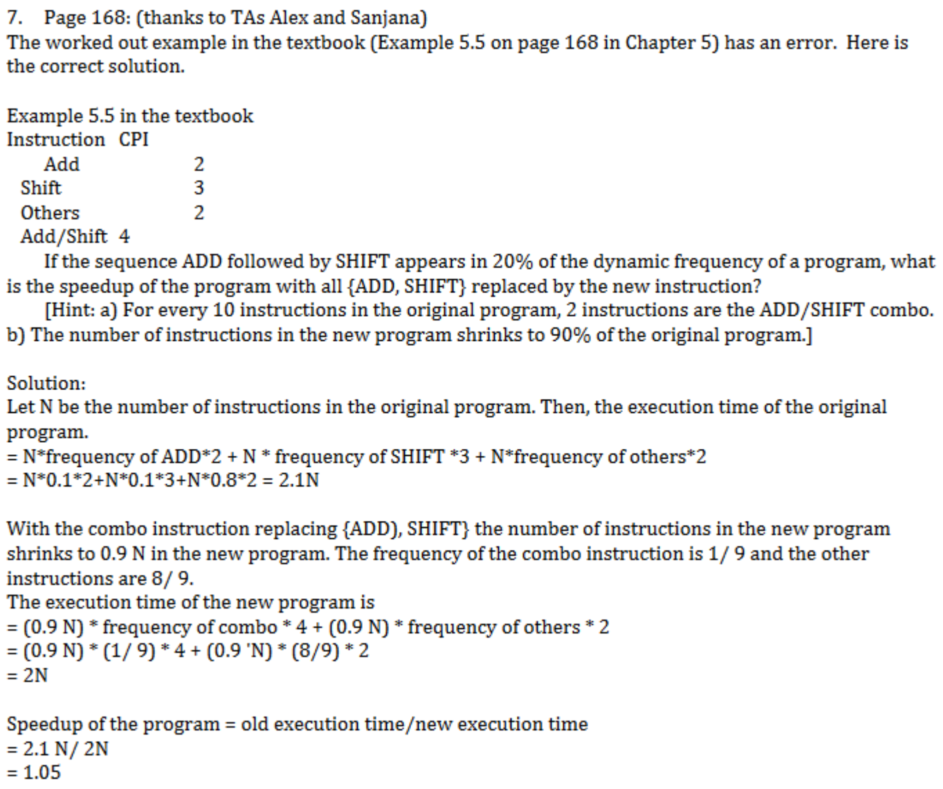
Does LW only have the "write-back" data to the destination register, so we need Rx, and Opcode too?

What about SW? since nothing needs to be written back, we just need the Opcode?

Commented [36]: Yes you are right.
Copied from textbook "The MEM stage requires the D-MEM. If the instruction processed by this stage in a clock cycle is not LW or SW, then the contents of the input buffer is simply copied to the output buffer at the end of the clock cycle. For LW instruction, the output buffer of the stage contains the D-MEM contents read, the opcode, and the destination register specifier (Rx ...

Commented [37]: Eg if we have an instruction like beq, it doesnt write back at all but it still copies from the previous stage? Why wouldnt it be like SW, and only have the opcode?

Commented [38]: The copy of EBUFF will be opcode. Since we need nothing to write in memory, there will be only opcode in EBUFF. They basically copy everything from EBUFF to MBUFF



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?
[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

Question 2. Branch prediction (10 points) (5 minutes)

We have modified the LC-2200 ISA to contain a new conditional branch instruction, **BNEQ**. When this instruction is executed, we branch to a target location if the two register operands are *not* equal. Assume we have a classical five-stage pipeline with all possible data forwarding paths. The pipeline is not equipped with any form of branch prediction and the ISA does not support branch delay slots. Given the following:

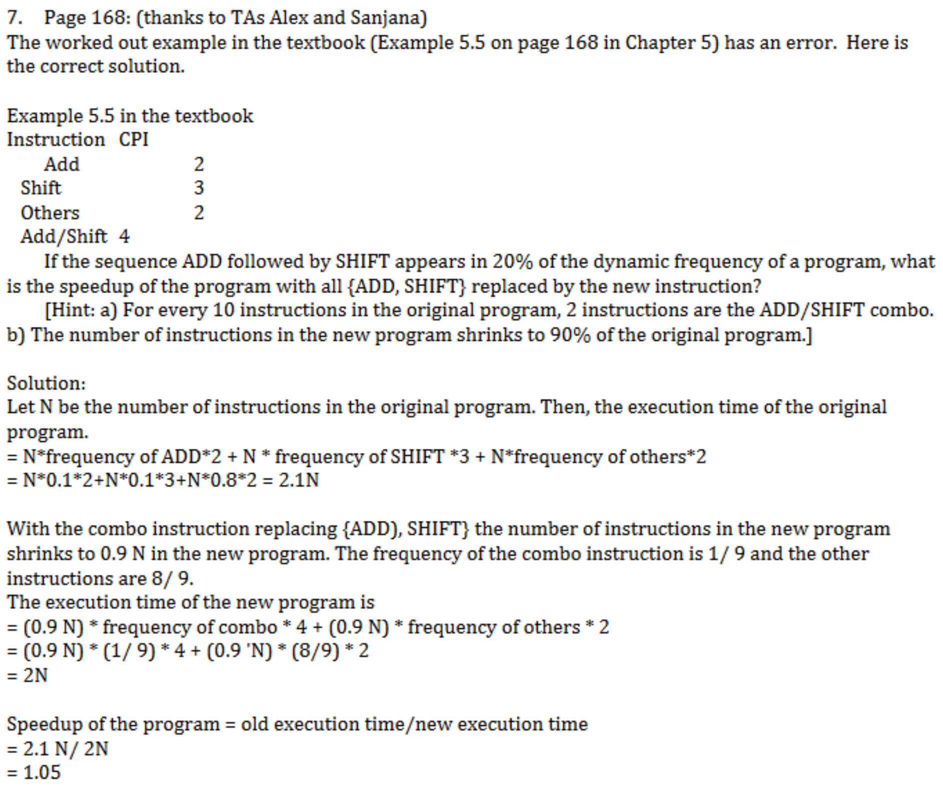
	0	1	2	3
0	t1=6			
1	t0=4			
2	t0=3	t0=2	t0=1	t0=0
3	var=3	var=2	var=1	var=0
4	neq	neq	neq	eq
5				v0=xf

Commented [39]: Do we assume the conservative approach is used here?

Commented [40]: I guess we do since we do not use both branch prediction and branch delay

Commented [41]: Don't we assume data forwarding? is available? the conservative approach would defeat the purpose of our data forwarding, no?

Commented [42]: wait nvm.



7. Page 168: (thanks to TAs Alex and Sanjana)

The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:

Let N be the number of instructions in the original program. Then, the execution time of the original program.

= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2

= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.

The execution time of the new program is

= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2

= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2

= 2N

Speedup of the program = old execution time/new execution time

= 2.1 N/ 2N

= 1.05

```
        lea $t1, var
        addi $t0, $zero, 4
loop:   addi $t0, $t0, -1
        sw $t0,0($t1)
        bneq $t0, $zero, loop
        addi $v0, $zero, 0xf
var:     .fill 0
```

a) Calculate the Cycles Per Instruction (CPI) achieved by the pipeline without any branch prediction. Explain your answer for credit. (6 points)

Cycle Number	IF	ID/RR	EX	MEM	WB

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

1	lea0-0				
2	addi0-1	lea0-0			
3	addi0-2	addi0-1	lea0-0		
4	sw0-3	addi0-2	addi0-1	lea0-0	
5	bneq0-4	sw0-3	addi0-2	addi0-1	lea0-0
6	addi0-5	bneq0-4	sw0-3	addi0-2	addi0-1
7	addi0-5	nop	bneq0-4	sw0-3	addi0-2
8	addi1-2	nop	nop	bneq0-4	sw0-3
9	sw1-3	addi1-2	nop	nop	bneq0-4
10	bneq1-4	sw1-3	addi1-2	nop	nop

Commented [43]: Is this not RAW and WAW?

Commented [44]: Do we have pass back values?

Commented [45]: it is RAW hazard but the pipeline supports data forwarding. so no bubbles

Formatted Table

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

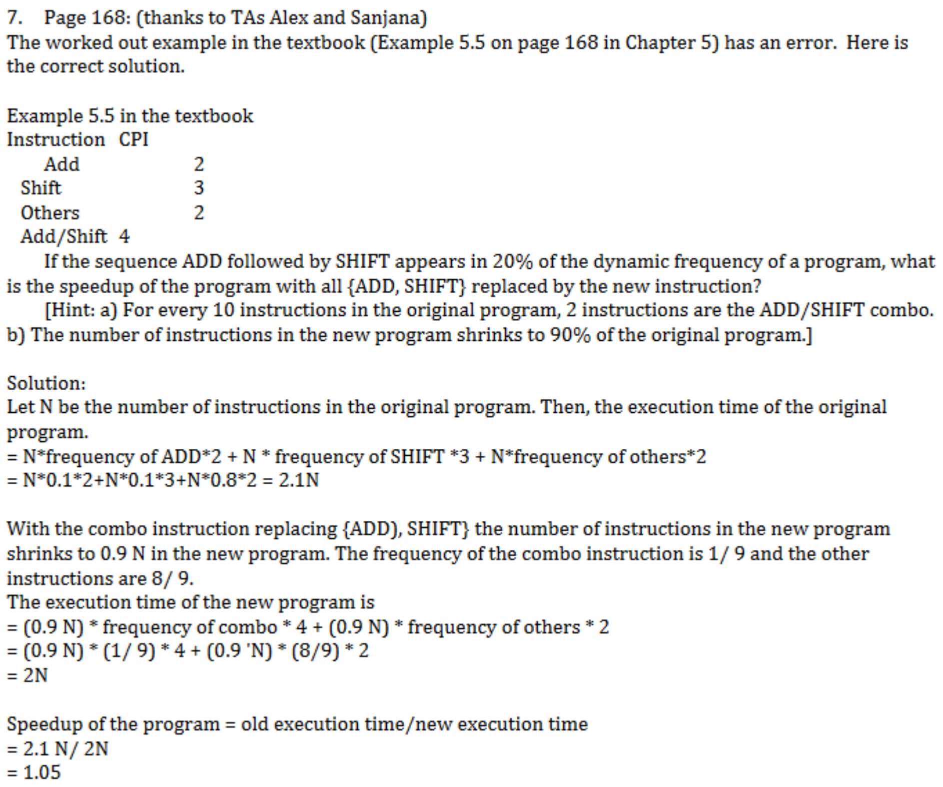
Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

11	addi1-5	bneq1-4	sw1-3	addi1-2	nop
12	addi1-5	nop	bneq1-4	sw1-3	addi1-2
13	addi2-2	nop	nop	bneq1-4	sw1-3
14	sw2-3	addi2-2	nop	nop	bneq1-4
15	bneq2-4	sw2-3	addi2-2	nop	nop
16	addi2-5	bneq2-4	sw2-3	addi2-2	nop
17	addi2-5	nop	bneq2-4	sw2-3	addi2-2
18	addi3-2	nop	nop	bneq2-4	sw2-3
19	sw3-3	addi3-2	nop	nop	bneq2-4
20	bneq3-4	sw3-3	addi3-2	nop	nop
21	addi3-5	bneq3-4	sw3-3	addi3-2	nop
22	addi3-5	nop	bneq3-4	sw3-3	addi3-2

Commented [46]: if the branch is "wrong", then the instruction in IR from 12 becomes a NOP --> 2 NOPs per loop



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

23		addi3-5	nop	bneq3-4	sw3-3
24			addi3-5	nop	bneq3-4
25				addi3-5	nop
26					addi3-5

num clock cycle=26

num instruction=15

Average CPI=26/15=1.73 to her

A more effective way of calculating number of cycles: (Joe)

Num of Instructions(assuming no bubbles and the pipeline is full) + bubbles + 4(for the last instruction to finish. Each instruction needs 5 cycles to complete. All instructions are counted once in the number of instructions(first term), so the last instruction has 4 cycles left(already counted 1))

Commented [47]: lmk what you think

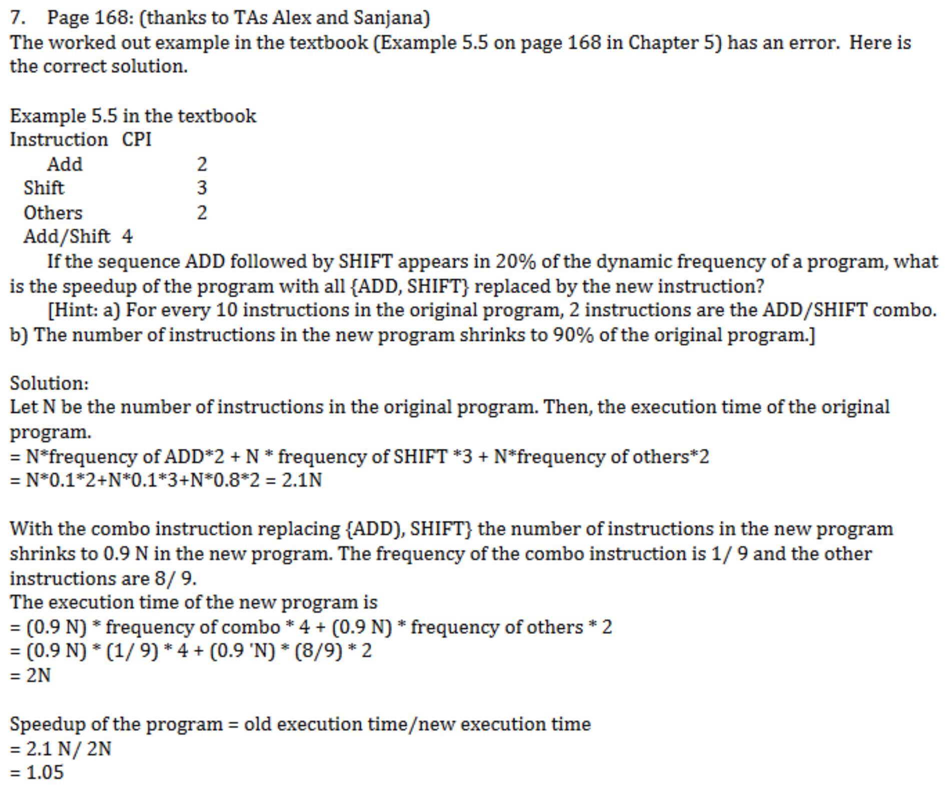
Commented [48]: I think what you wrote is a correct and more concise reasoning. Essentially we treat nops as instructions. Therefore, there will be a total of 7 nops produced in total (2 nops for 3 branch taken, 1 nop for no branch taken). Therefore, we have executed 7 more instructions in addition to the 15 that were written down and useful. Also, it takes additionally 4 clock cycles for the last instruction to retire. Therefore, the total is 15 + 7 + 4 = 26 cycles. Part B can be reasoned in a similar manner. The penalty of mis-prediction is 2 cycles, meaning 2 nops. The last branch was predicted correctly, and we have one less nop compared to part A. Hence, the total number of instructions exceuted can be viewed as 15 + 6 = 21. However, it takes four additional cycles for the last instruction to retire, so the cycle count is 25.

Commented [49]: who's joe?

Commented [50]: Im joe

Commented [51]: Hi Joe

Commented [52]: Hi there



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

= 15 + 2*3(branch) + 1(no branch) + 4 = 26

b) Now assume we extend the pipeline’s hardware to always predict that the branch is not taken and calculate the CPI again. Explain your answer for credit. (4 points)

BNEQ makes the same number of bubbles as stalling the pipeline if it is mispredicted. So, there will be differences only in the last bneq instruction. With branch prediction, the last bneq will not make a bubble and the total number of clock cycles would be 25 which is one less than stalling the pipe method. So, average cpi would be 25/15=1.67

Question 3. Pipeline Depth (9 Points) (6 minutes)

a) The classical pipeline we studied in has a total of five stages. Modern processors, however, can have many more (upwards of 15 stages!). What is the benefit of a deeper pipeline compared to a shorter one? (3 points)

Commented [53]: So we have 2 bubbles per branch taken to flush and 1 bubble per branch not taken? I thought data forwarding means no bubbles?

Commented [54]: That is for structural hazards, not control hazards

Commented [55]: data forwarding helps you remove RAW bubbles, but you still get bubbles from the control hazard, which is essentially conditional branching

Commented [56]: Structural hazards are dealt usually through increasing more hardware components. By the question prompt, it should be assumed that all structural hazards are handled by the pipeline's design, such as additional ALUs for EX stage and IF stage.

Commented [57]: I think MinGeun meant data hazards instead of structural hazards

Commented [58]: why is this 4

Commented [59]: it's for the last instruction (addi) to finish executing

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

I think the most important thing to mention here is that more stages means the clock speed can be higher which means more throughput. I think that's what they wanna see.

(5-56) A deeper pipeline can have multiple decode and functional units, which can be useful for multiple issue processing, allowing a processor to issue and execute multiple instructions at once independently while also maintaining ILP. **(I couldn't find anything else on benefit, can someone else help me elaborate on here?) increases throughput?**

I agree. Throughput is # of instructions per clock cycle thus more instructions per clock cycle means more throughput. Additionally, you can utilize more resources (Regfile reads/writes, memory reads/writes, etc.) with more stages.

1. Breaking up the need for multiple cycles to carry out a specific operation into multiple stages is one way of dealing with the complexity.

2. The EX unit of the simple 5-stage pipeline gets replaced by a collection~~by collection~~ of functional units. There may be an additional stage to help schedule these multiple functional units.

Commented [60]: What's ILP again?

Commented [61]: instruction level parallelism

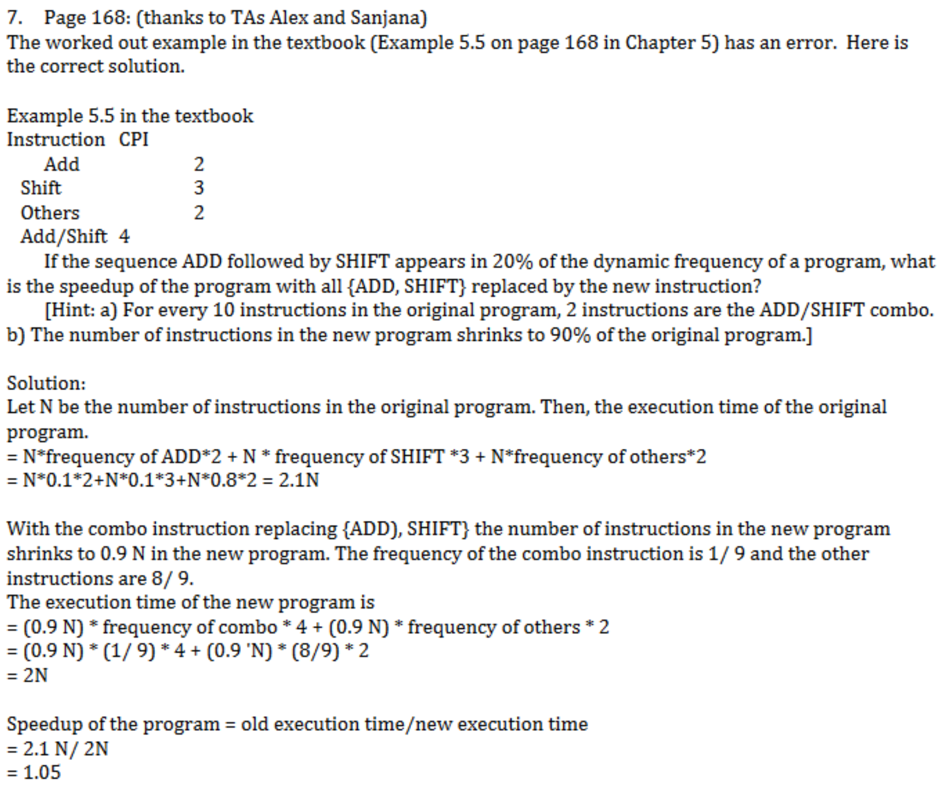
Commented [62]: What do yall think?

Commented [63]: look good to me

Commented [64]: Technically the throughput will remain the same (since there will always be 1 instruction being executed per clock cycle), but the clock cycle speed will increase, so each instruction will execute in less time (not necessarily less clock cycles)

Commented [65]: why does more stages = higher clock cycle speed?

Commented [66]: I believe it's because you can break down the work into smaller pieces, each stage has less to do



7. Page 168: (thanks to TAs Alex and Sanjana)

The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:

Let N be the number of instructions in the original program. Then, the execution time of the original program.

= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2

= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.

The execution time of the new program is

= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2

= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2

= 2N

Speedup of the program = old execution time/new execution time

= 2.1 N/ 2N

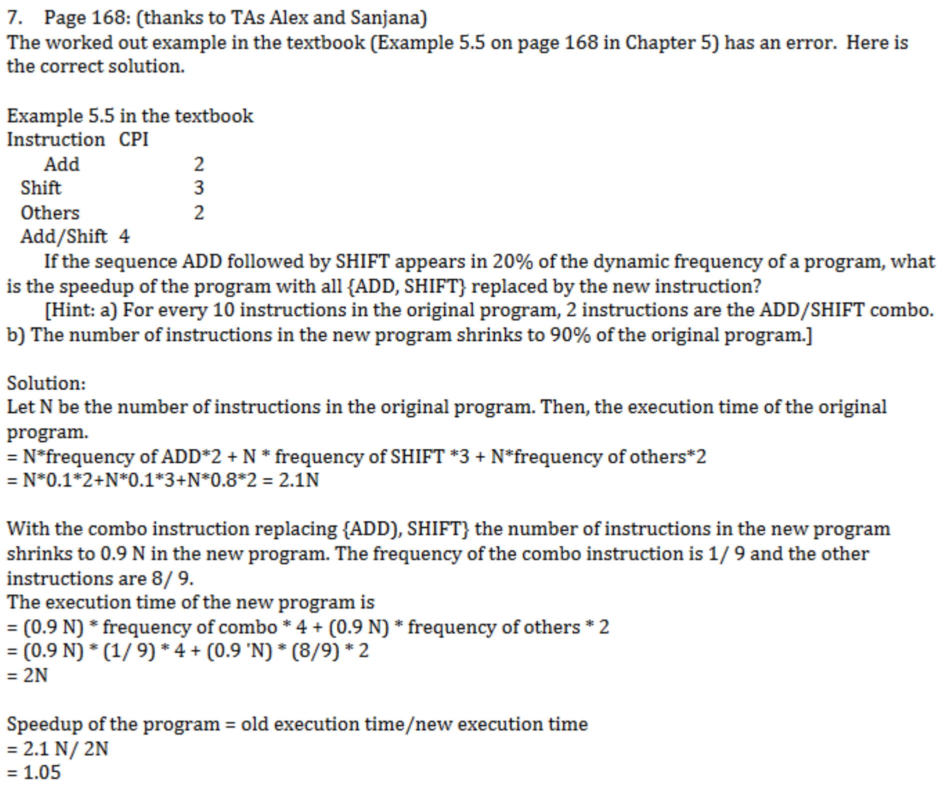
= 1.05

3. The processor may use an additional stage to detect resource conflicts and take actions to disambiguate the usage of registers using a technique referred to as register renaming.

4. Both ROB and/or register renaming (in hardware) help hardware-based speculation since the information in the physical registers or ROB used in the speculation can be discarded later when it is determined that the effects of a speculated instruction should not be committed.

Another thing that is important: we want each stage to do approximately the same amount of work so that we aren't waiting on anything to finish all the time. Having lots of stages means we can more easily customize the amount of work done by each stage, and make this balance easier to achieve.

The textbook says this about "deep pipelines:" Typically, the microarchitecture includes specialized functional units for performing floating-point add, multiply, and division that



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

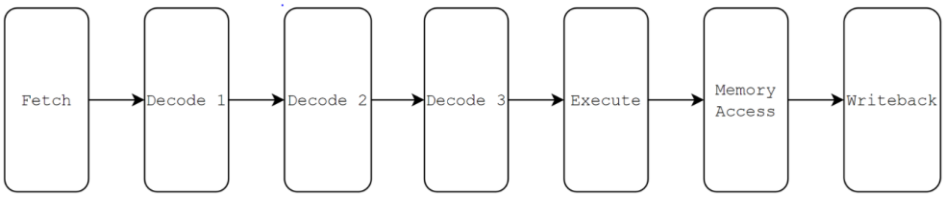
Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

are distinct from the integer ALU. The EX unit of the simple 5-stage pipeline gets replaced by this collection of functional units (see Figure 5.20). There may be an additional stage to help schedule these multiple functional units.

b) Say that we modified the LC-2200 pipeline to look like the following figure. Assume the pipeline uses branch prediction and a branch that is *not* taken is mispredicted as taken.



How many bubbles would result in the pipeline? Explain your answer. (3 points)

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

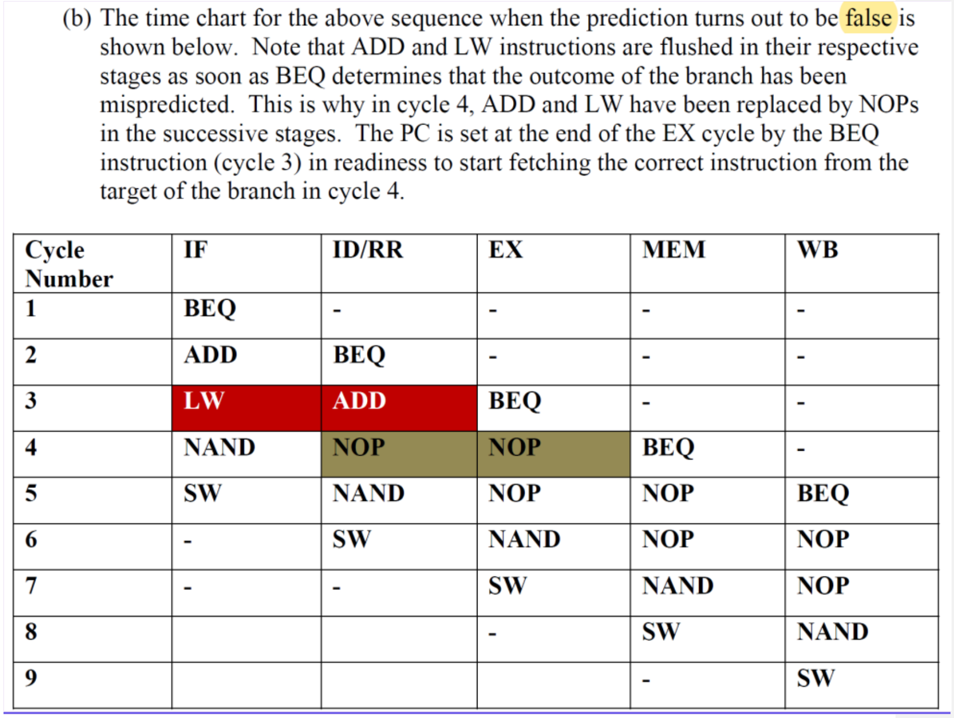
If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

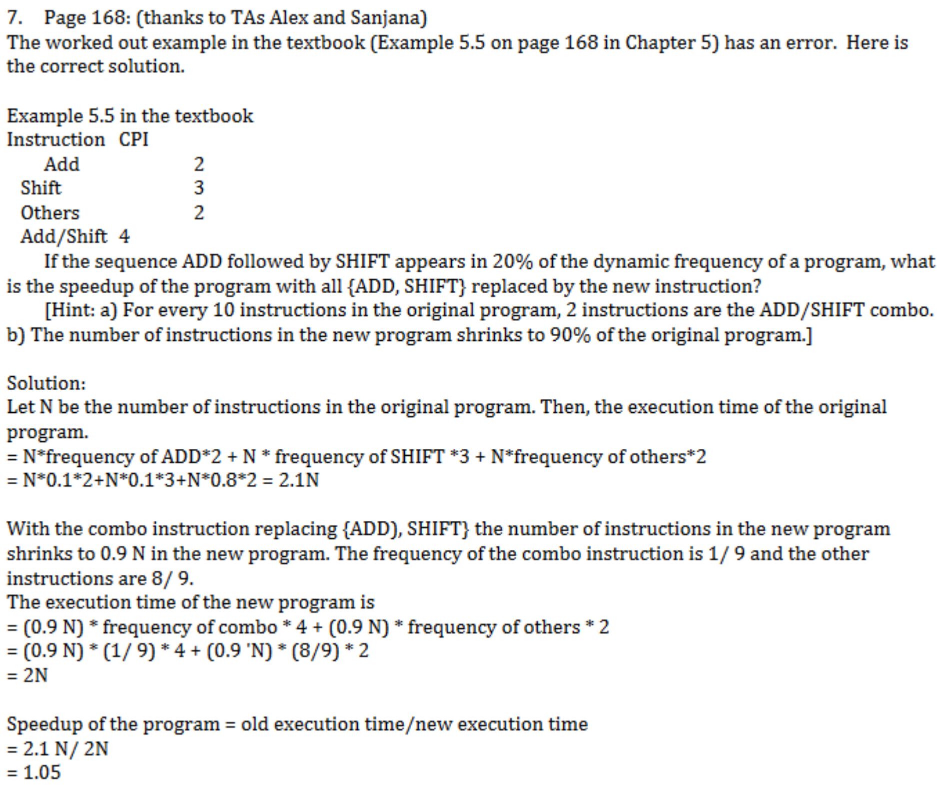
With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05



(b) The time chart for the above sequence when the prediction turns out to be false is shown below. Note that ADD and LW instructions are flushed in their respective stages as soon as BEQ determines that the outcome of the branch has been mispredicted. This is why in cycle 4, ADD and LW have been replaced by NOPs in the successive stages. The PC is set at the end of the EX cycle by the BEQ instruction (cycle 3) in readiness to start fetching the correct instruction from the target of the branch in cycle 4.

Cycle Number	IF	ID/RR	EX	MEM	WB
1	BEQ	-	-	-	-
2	ADD	BEQ	-	-	-
3	LW	ADD	BEQ	-	-
4	NAND	NOP	NOP	BEQ	-
5	SW	NAND	NOP	NOP	BEQ
6	-	SW	NAND	NOP	NOP
7	-	-	SW	NAND	NOP
8			-	SW	NAND
9				-	SW



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?
[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

Br						
I1	Br					
I2	I1	Br				
I3	I2	I1	Br			
I4	I3	I2	I1	Br		
After Br	NOP	NOP	NOP	NOP	Br	

4 bubbles. Since the prediction turns out to be false after execution, instructions in Decode1, Decode 2, Decode 3, and Execute~~Fetch, Decode1, Decode2, and Decode3~~ will be flushed. So there will be four bubbles.

c) Comment on a potential disadvantage of a longer pipeline on the performance of the currently executing program, other than an increased branch penalty. (3 points)

Hazards

Commented [67]: do all the instructions have to pass through all the decode?

Commented [68]: I'm not sure. But I saw instructions before execution stage were flushed in the textbook.

Commented [69]: Can you point to me which part of the textbook it is?

Commented [70]: Added an image

Commented [71]: But this is different from the multi-decoder setting, no?

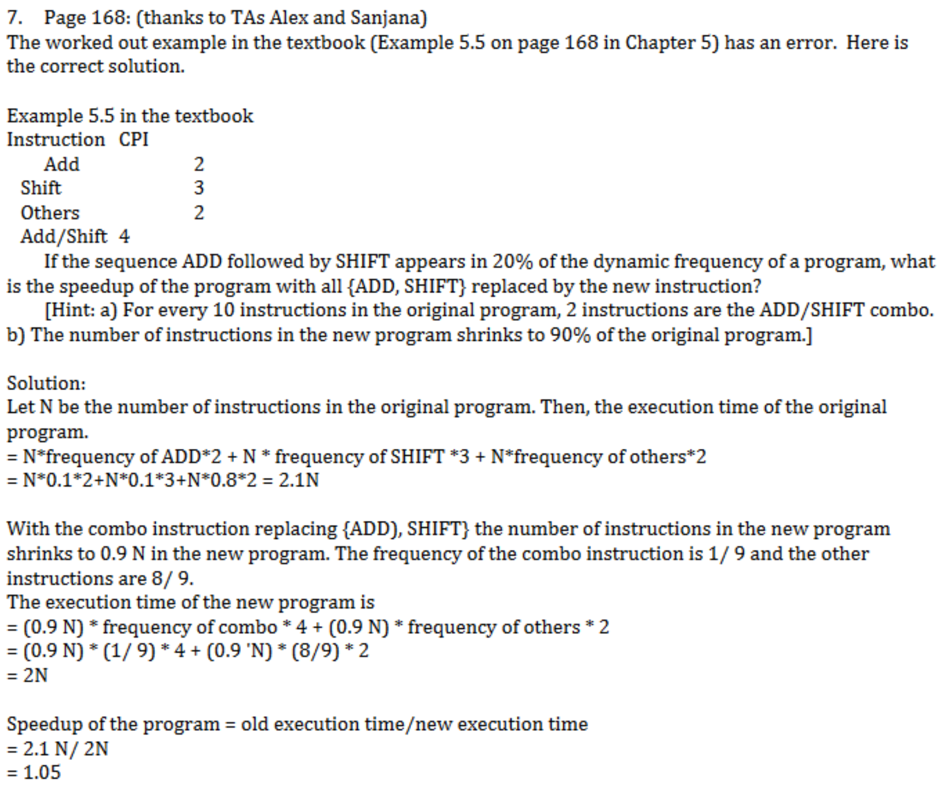
Commented [72]: Does the number of decoder matter? I mean ID/RR stage can contain multiple decoders in it. I'm not sure tho

Commented [73]: wait are we sure about this? because they were in fetch, decode 1, decode 2, decode 3 when the branch is in EX. they become nops in the next cycle when the pipeline is flushed and the ...

Commented [74]: Does each decode serve different purposes? How do we know whether the instruction ...

Commented [75]: If you think of SW, we do nothing at WB stage but SW instruction always pass through W ...

Commented [76]: makes sense. thanks!



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

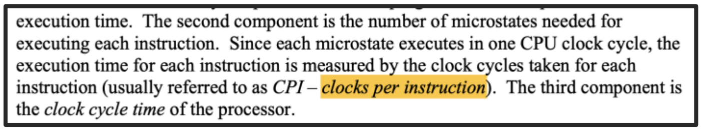
Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

If there is no data forward supporting the pipeline, there will be more latency needed to deal with data hazard problems. Due to this, CPI will be increased

Commented [77]: wouldn't decreased cpi be a bonus for better throughput? Also, pretty sure that when hazards appear, the CPI would increase, no?

Commented [78]: I agree, lower cycles per instruction should mean higher efficiency

Commented [79]: more clock cycles per instruction lead to lower efficiency



execution time. The second component is the number of microstates needed for executing each instruction. Since each microstate executes in one CPU clock cycle, the execution time for each instruction is measured by the clock cycles taken for each instruction (usually referred to as *CPI – clocks per instruction*). The third component is the *clock cycle time* of the processor.

1. Longer latency
2. Harder to keep all pipes filled unless you have plenty of parallelism
3. Any structural hazards if multiple stages in pipeline needs to be accessing shared resources
4. ^^^^ From Quora lol

Question 4. Data Hazards (9 Points) (8 minutes)

Consider the following snippet of LC-2200 assembly code:

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

```
I1: ADD $t0, $t1, $t2
I2: ADD $t1, $t0, $t1
I3: LW  $t2, 0($t2)
I4: ADD $t0, $t2, $t1
```

a) For each instruction, identify any of the three types of data hazards (RAW, WAR, WAW) that exist between all instruction pairs (e.g., for I4, you need to consider potential hazards with instructions I1, I2, and I3). (3 points)

	I1	I2	I3	I4
I1	[redacted]	[redacted]	[redacted]	[redacted]
I2	RAW, WAR	[redacted]	[redacted]	[redacted]
I3	WAR	None	[redacted]	[redacted]

Commented [80]: Why is this WAR instead of RAW?

Commented [81]: Because we write $t2 at I3 after we read $t2 at I1

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

I4	WAW	WAR, RAW	RAW	[redacted]

RAW in I2: You are reading $t0 before it has been written from I1

~~b)~~ Assuming the classical five-stage pipeline without any data forwarding support, count the number of bubbles that will be present upon the execution of the program. Explain your answer for credit. (4 points)

b)

add t0, t1, t2
THIS TABLE HAS BEEN UPDATED, IT SHOULD BE CORRECT NOW

Cycle #	IF	ID/RR	EX	MEM	WB	COMMENTS
1	I1	-	-	-	-	
2	I2	I1	-	-	-	

Commented [82]: I believe that WAW occurs when there is no "read" in between the two writes. But in this case, we have I2 reading it. But I am not sure whether the question asks us to assess the hazards independently (meaning ignoring other instructions) or not :/

Commented [83]: I agree. From "5.13.2.4 Other types of Data Hazards", WAW essentially means that a register value that was written by one instruction is being over-written by a subsequent instruction with no intervening read of that register.

Commented [84]: yea but we dont know whether they ask us to evaluate processes independently of others

Commented [85]: Any thoughts?

Commented [86]: They havent said anything as far a

Commented [87]: it irritates me that they haven't. At

Commented [88]: My best guess is that WAW shoul

Commented [89]: I agree

Commented [90]: I think everyone's stuck on this.

Formatted: Font color: Black

Formatted

7. Page 168: (thanks to TAs Alex and Sanjana)

The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:

Let N be the number of instructions in the original program. Then, the execution time of the original program.

= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2

= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.

The execution time of the new program is

= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2

= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2

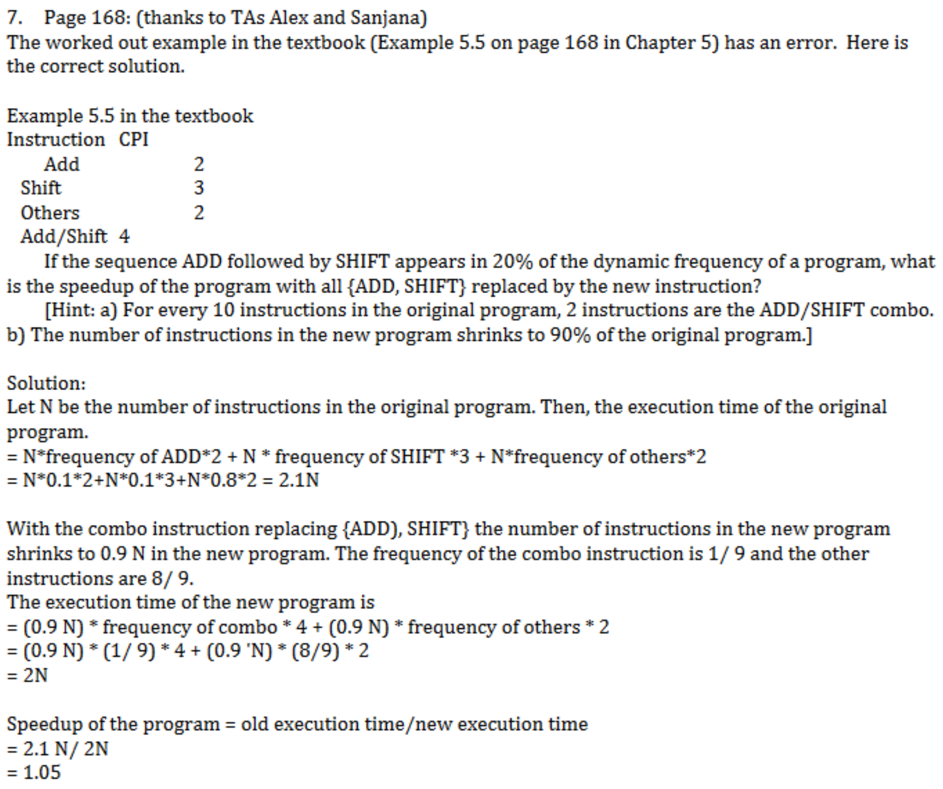
= 2N

Speedup of the program = old execution time/new execution time

= 2.1 N/ 2N

= 1.05

3	I3	I2	I1	-	-	I2 sees that $t0’s write pending bit is set, so it can’t proceed (to EX) until that is gone
4	I3	I2	NOP	I1	-	Need to wait for I1 to write before executing I2
5	I3	I2	NOP	NOP	I1	Same as above
6	I3	I2	NOP	NOP	NOP	Only once I1 is gone can we continue
7	I4	I3	I2	NOP	NOP	I3 doesn’t depend on I2, so we’re good to continue
8	-	I4	I3	I2	NOP	
9	-	I4	NOP	I3	I2	Need to wait for I2 to be out of pipeline before continuing (both use $t1)
10	-	I4	NOP	NOP	I3	See above



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

11	-	I4	NOP	NOP	NOP	I4 must wait until I3 has written to register, so until it is out of the pipeline
12	-	-	I4	NOP	NOP	
13	-	-	-	I4	NOP	
14	-	-	-	-	I4	I4 finishes without problem

TOTAL # OF BUBBLES = 3(I1&I2) + 2(I2&I4) + 1(I3&I4, because there are already 2 bubbles after I3 because I2 needs to stall I4) = 6 (feel free to edit if you find mistakes)

Commented [91]: So the number of bubbles I agree with, but why do we explain it like that? Isn't it just as valid to say that I3 is responsible for all 3 bubbles or vise versa? how do we know which bubbles are being made by which instructions?

Commented [92]: I agree with you, I think your explanation also works. So long as you provide a valid explanation, I'm sure they'll accept it.

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

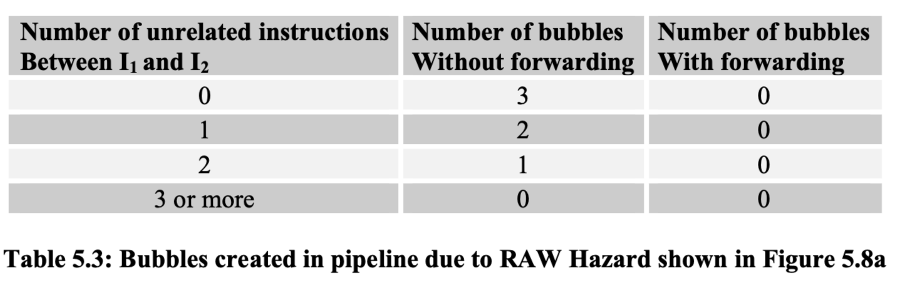
If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05



Number of unrelated instructions Between I_1 and I_2	Number of bubbles Without forwarding	Number of bubbles With forwarding
0	3	0
1	2	0
2	1	0
3 or more	0	0

Table 5.3: Bubbles created in pipeline due to RAW Hazard shown in Figure 5.8a

c) Count the number of bubbles with full data forwarding support. Explain your answer for credit. (2 points)

Data forwarding means zero bubbles* unless you get data from the memory(**someone please confirm this thx**), but since we have an LW instruction, it requires **one** bubble.

Commented [93]: Won't one bubble still be needed to handle the RAW hazard between I3 and I4? since I3 is a LW instruction, the value of $tw won't be available till MEM stage

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

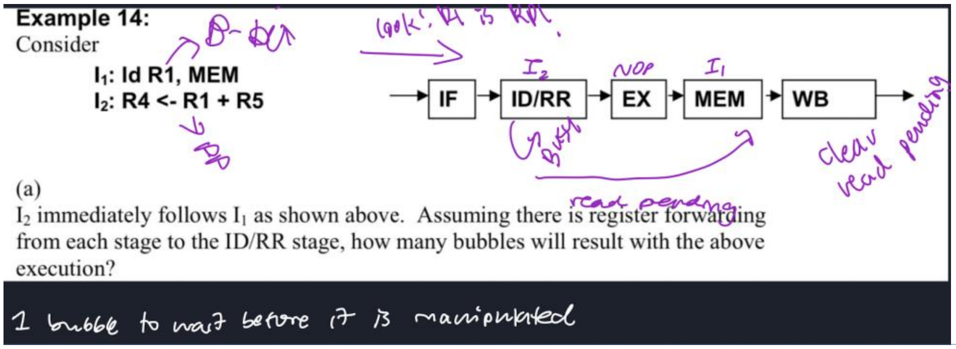
If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05



Example 14:
Consider

I_1: ld R1, MEM
I_2: R4 <- R1 + R5

(a)
I_2 immediately follows I_1 as shown above. Assuming there is register forwarding from each stage to the ID/RR stage, how many bubbles will result with the above execution?

1 bubble to wait before it is manipulated

Commented [94]: yea its one bubble i believe. 0 for all the others but the LW needs 1 bubble in a RAW hazard even with forwarding (see page 197 in textbook)

Commented [95]: lw would still potentially have one bubble (Table 5.4 in the textbook)

Commented [96]: Agree

Commented [97]: LW has 1 bubble since it gets its value during the Mem State, and since I4 immediately precede I3 and needs the value in the ID/RR state, a NOP has to exist and occupy the EX state so that the LW is able to get to MEM and forward its data to I4 which is currently in ID/RR

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

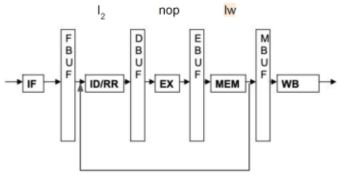
Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

Exception

Can we forward the data in EX stage if it is a load instruction?



No, we must wait for an extra clock cycle so it can get the value in MEM stage.

Pls check on me. Might have misheard during lab!

The load instruction will only have something from memory to forward at the completion of the MEM state, which means that there needs to be a NOP to stall until the MEM phase completes.

NOTE: this only matters if there is a LW instruction, and the instruction IMMEDIATELY after uses the same register as the destination of the LW

This means that the total number of bubbles in the pipeline with forwarding support is 1

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

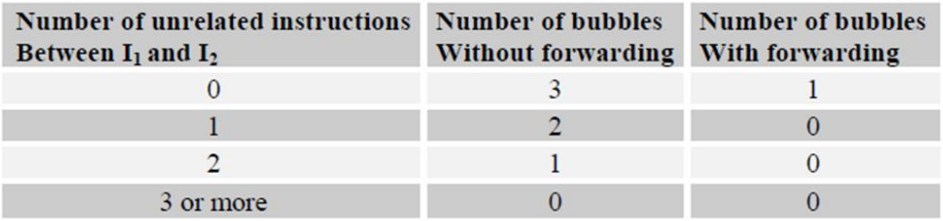
If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

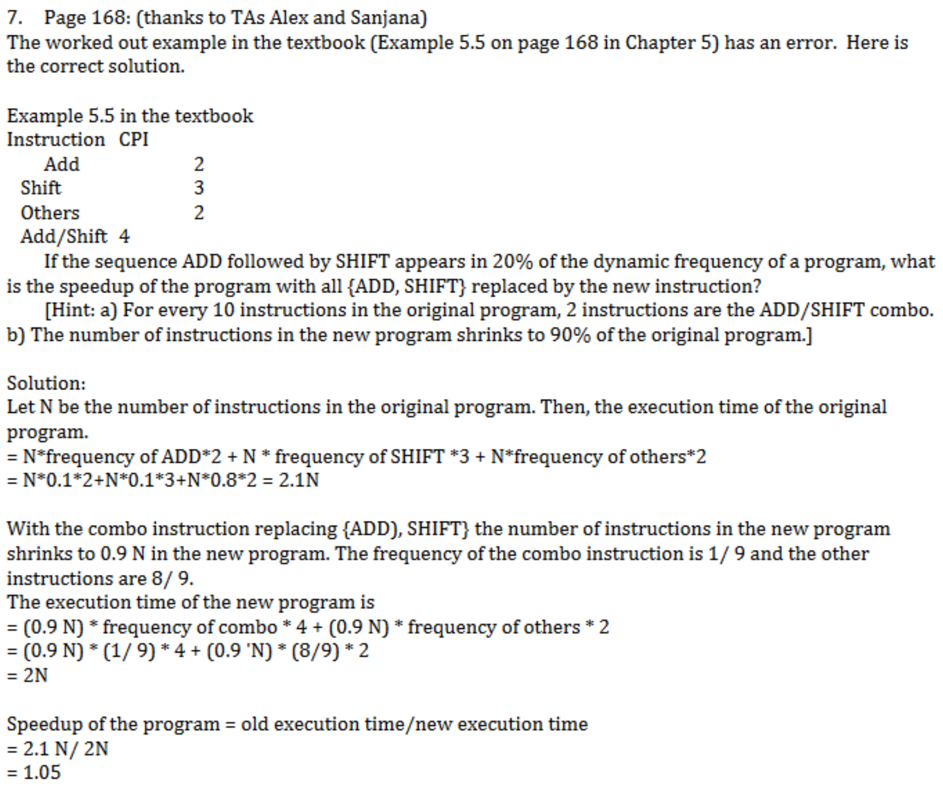
Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05



Number of unrelated instructions Between I_1 and I_2	Number of bubbles Without forwarding	Number of bubbles With forwarding
0	3	1
1	2	0
2	1	0
3 or more	0	0

Table 5.4: Bubbles created in pipeline due to Load instruction induced RAW Hazard

	IF	ID/RR	EX	MEM	WB
1	I1				
2	I2	I1			
3	I3	I2	I1		



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

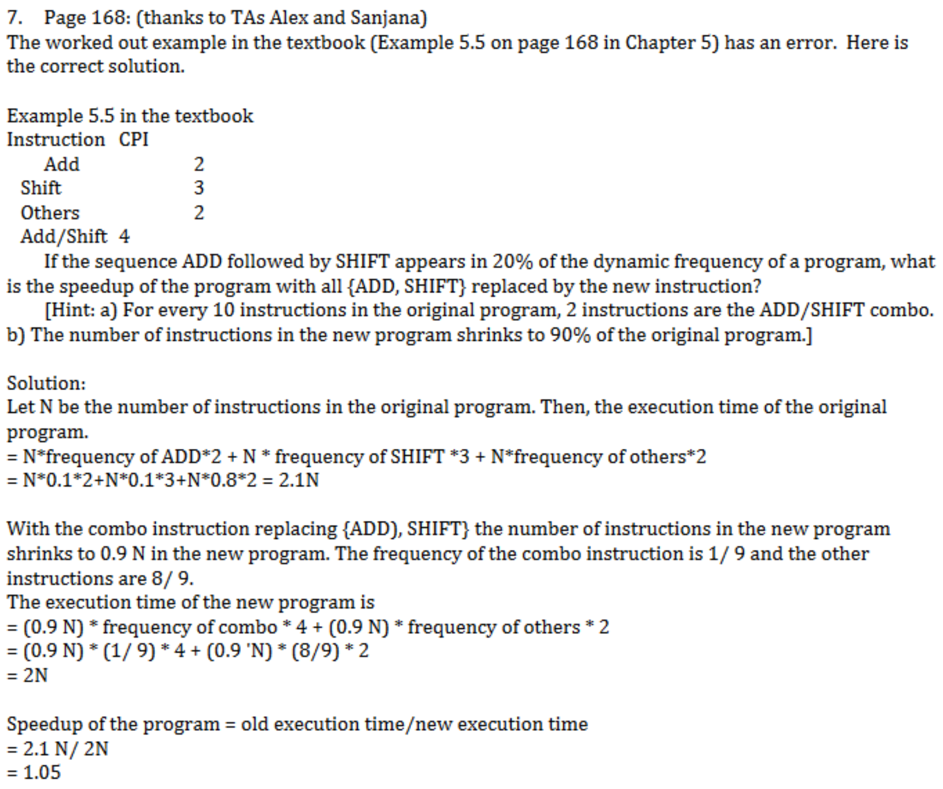
With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

4	I4	I3	I2	I1	
5		I4	I3	I2	I1
6		I4	NOP	I3	I2
7			I4	NOP	I3
8				I4	NOP
9					I4

There is a RAW hazard between I3 and I4. Also, I3 includes LW instruction, we need **one bubble** since we need to stall until the MEM stage completes.

Question 5. Branch Table Buffer (9 Points) (5 minutes)



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

Consider a 5-stage pipelined processor (same as the one in the textbook) equipped with a branch target buffer (BTB which has only 1 bit of branch history). Assume a 32-bit word-addressable architecture. Upon misprediction, there will be a 2-cycle penalty (i.e., two NOPs). A BEQ instruction is fetched from memory location 1000 by the "IF" stage. The BTB currently has an entry for this BEQ instruction:

```
PC=1000 | branch predicted TAKEN;

PC of BEQ target = 1200;

PC of next sequential instruction = 1001
```

a) What will be the action in IF stage in the next clock cycle? (2 points)
   i) it should fetch whatever it predicts will be the next instruction. We don't know the outcome of the BEQ until the EX state, which hasn't happened yet. While the processor waits for that to happen, it decides to fetch and decode the instruction it predicts will happen, which is whatever is at address 1200.

b) When BEQ is in the EX stage, the outcome of the branch turns out to be NOT TAKEN. What are the actions that will ensue as a result of this outcome? (7 points)

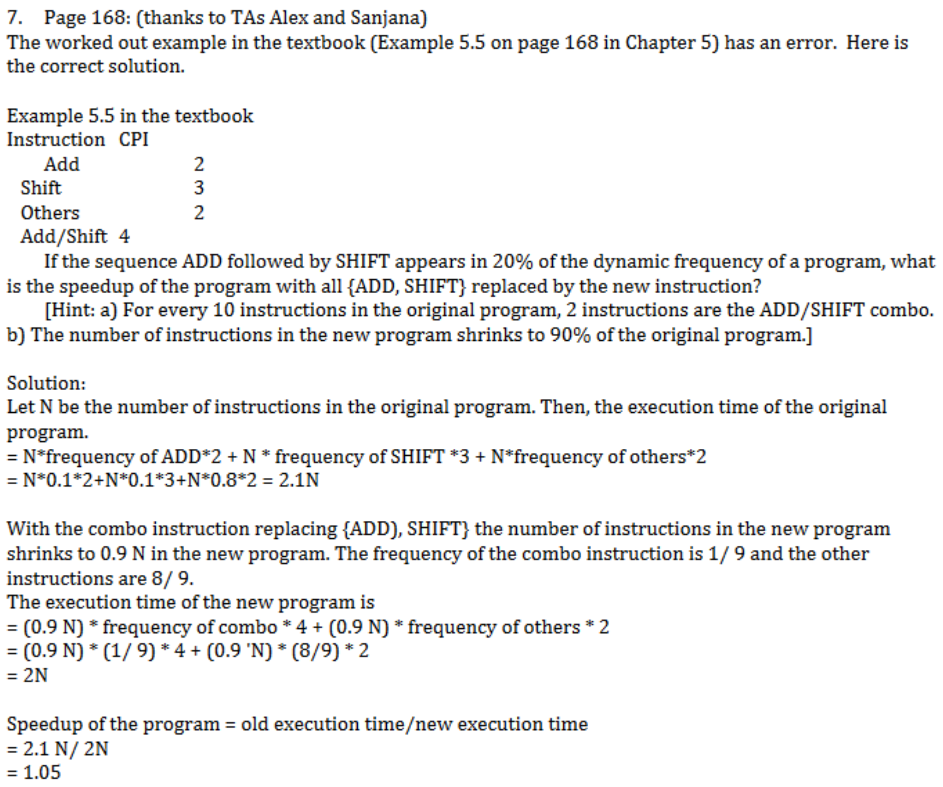
Commented [98]: Is this saying that it will fetch the instruction at 1200 or 1001 because the wording implies you are suggesting 1001 when you later say 1200

Commented [99]: It will be 1200 as the branch is predicted as taken

Commented [100]: Since it asks only about the next clock cycle, we probably can't say decode. Just that it's fetched

Commented [101]: Agreed

Commented [102]: Agreed



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

i) There would be 2 bubbles in the pipeline when there did not need to be because the next sequential instruction would be at the IF stage when BEQ is at MEM (i think?)

ii) Yes, once the processor realizes it predicted wrong, it will turn the two operations before it into NOP and its next fetch will come from the desired location. So it will look like:

Instruction @ 1001 -> NOP -> NOP -> BEQ -> Whatever came before the BEQ

The BTB will also be updated to reflect the failed prediction

Commented [103]: I agree with this

Commented [104]: I think it also updates the BTB to invalidate the last prediction result, no?

Commented [105]: This is correct, yeah

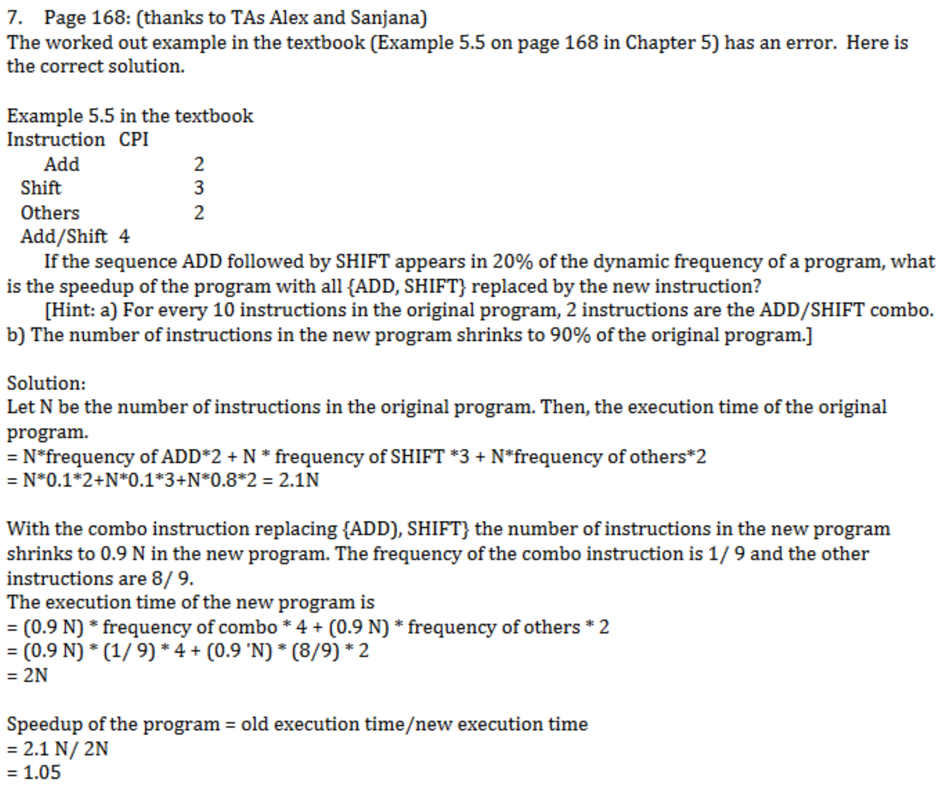
Commented [106]: Does this mean (IF) = 1001, (ID/RR) = NOP, (EX) = NOP, (MEM) = BEQ, (WB) = what action comes before BEQ?

Commented [107]: For 1 bit BTB, is it going to change?

Commented [108]: I would assume so, as the most recent result is now "branch not taken" for that branch instruction, so we need to update the BTB to represent that.

Commented [109]: I think the 1 bit just indicates whether branching happened last time or not

Commented [110]: Yeah p much (which is why you update it)



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?
[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

Question 6. Scheduling (12 Points) (8 minutes)

Shown below are the duration of the CPU burst and I/O burst times for the three processes.

	P1	P2	P3
CPU	2	6	3
I/O	4	1	3

Assume processes P1, P2, and P3 are ready to run at the beginning of time. Assume each process does:

- CPU burst; first CPU burst
- I/O burst; one I/O burst
- CPU burst; second CPU burst
- Done; process execution complete

Priorities:

Commented [111]: Priorities only come into place for tie-breaking right?

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

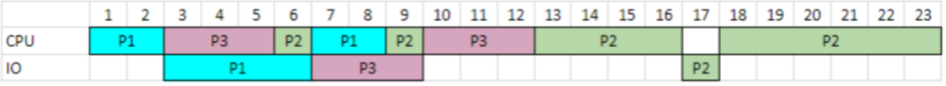
Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

- P3 highest priority
- P2 next higher priority
- P1 lowest priority

Process arrival:

- P1, P2, P3 all arrive within milliseconds of each other in this order
- all are ready to be scheduled at time 0

Shown below are the timelines of activities on the CPU and I/O for the three processes with some scheduling discipline.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
CPU	P1		P3			P2	P1		P2	P3			P2					P2					
IO			P1				P3										P2						

a) What scheduling algorithm is implemented here? Explain your choice. (3 points)
SRTF

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

At t=7, P1 cuts P2 off because both are in the ready queue and p1 has shorter remaining time(2 < 5). At t=10, P3 and P2 are in the ready queue and P3 gets serviced first because p3 has shorter remaining time(3<4)

b) Why does the scheduler pick P3 to run on the processor time 10? (3 points)

Once P3's IO Burst is done, it goes back to the CPU Burst. In this case, P3's CPU Burst Time currently (3) is less than P2's CPU Burst Time (6-2=4). Thus, P3 gets to run first.

c) What is the wait time for P2? Explain your answer. (3 points)

Total time = 23
Execution time = 13
w = t - e = 23 - 13 = 10

d) What is the turnaround time for P3? Explain your answer. (3 points)

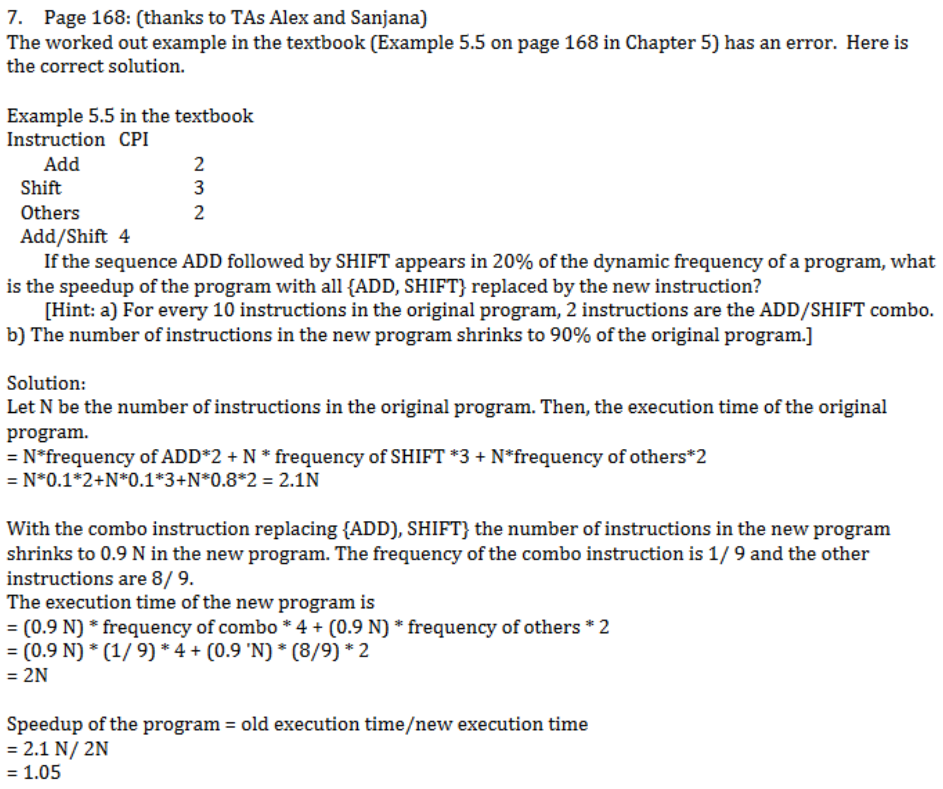
12 (The last CPU burst ends at time 12 and P3 is ready at time 0) 12-0= 12.

Commented [112]: Isnt this technically 6-2 = 4 + 6 = 10 (which is the next CPU burst) since SRTF looks at total remaining CPU time?

Commented [113]: it only counts for one round of CPU

Commented [114]: How was 13 calculated for the execution time here?

Commented [115]: The question says each process runs 1 CPU + 1 IO + 1 CPU, so 6+1+6=13



7. Page 168: (thanks to TAs Alex and Sanjana)

The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:

Let N be the number of instructions in the original program. Then, the execution time of the original program.

= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2

= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.

The execution time of the new program is

= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2

= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2

= 2N

Speedup of the program = old execution time/new execution time

= 2.1 N/ 2N

= 1.05

Question 7. Round Robin Scheduler (17 Points) (10 minutes)

We are writing a new scheduler that will implement a round robin algorithm in order to schedule processes on the processor. The ready queue contains three processes P1, P2, P3 in that order. The execution characteristics of the three processes are as shown in the table below.

Process ID	CPU Burst	I/O Burst	CPU Burst
P1	10	2	15
P2	4	2	3
P3	3	2	3

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

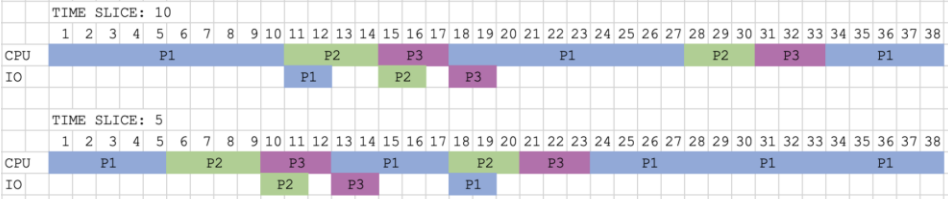
If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05



	TIME SLICE: 10																																					
	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	31	32	33	34	35	36	37	38
CPU	P1										P2				P3			P1										P2			P3			P1				
IO											P1				P2			P3																				
	TIME SLICE: 5																																					
	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	31	32	33	34	35	36	37	38
CPU	P1					P2				P3			P1					P2			P3			P1					P1					P1				
IO										P2			P3					P1																				

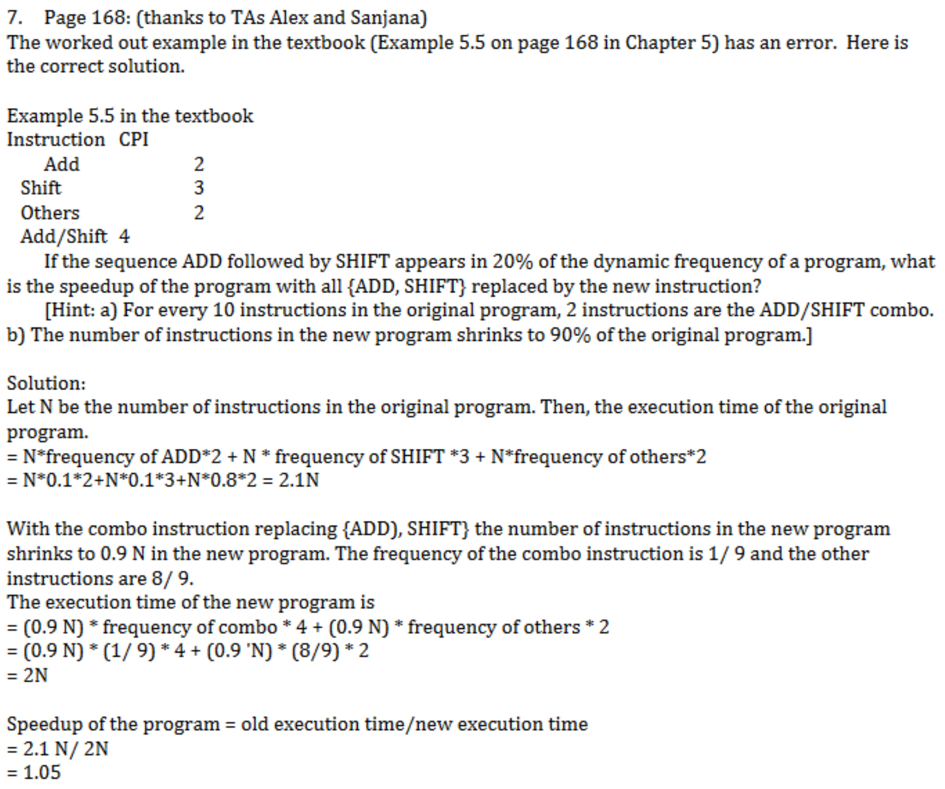
a) Given the above timeline with a timeslice of ten (10) time units, and assuming no scheduling or context-switching overhead, compute the average waiting time for the three processes. Show your work for credit. (3 points)

Avg wait = (w1 + w2 + w3) / 3, w = time in processor but not being executed

= (11 + 21 + 25) / 3

= 57 / 3 = 19

b) Given the above timeline with a timeslice of five (5) time units, and assuming no scheduling or context-switching overhead, compute the average waiting time for the three processes. Show your work for credit. (3 points)



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

Avg wait = (w1 + w2 + w3) / 3, w = time in processor but not being executed

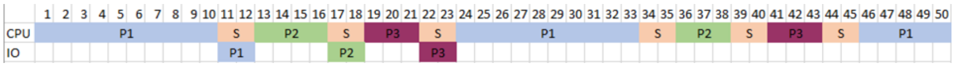
= (11 + 11 + 15) / 3

= (37 / 3) = 12.33

c) Now assume that the scheduling and context-switching overhead (i.e., the time to pick the next process to run and dispatch it on the processor) is two (2) time units and I/O completions do not introduce any additional overheads. Compute the average waiting time for the above processes, *for each of the two timeslice values* (5 and 10). Show your work for credit. (5 points)

NOT SURE MY FORMULA IS CORRECT, CAN SOMEONE CONFIRM?

I calculated the problem again and got (I think) better answers



	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	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
CPU	P1										S		P2				S		P3			S		P1										S		P2			S		P3			S		P1				
IO											P1						P2					P3																												

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

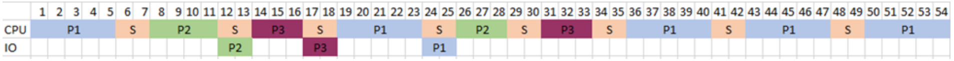
Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

Avg wait$_{10}$ = (w1 + 2(s1) + w2 + 2(s2) + w3 + 2(s3)) / 3,
w = time in processor but not being executed, S = # of switches made **before** the process in question completes

= (11 + 12 + 21 + 8 + 25 + 10) / 3

= (87) / 3

= 29



	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
CPU	P1					S		P2				S		P3			S		P1					S		P2	
IO												P2					P3							P1			

	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
CPU		S		P3			S		P1					S		P1					S		P1				
IO																											

Avg wait$_{5}$ = (w1 + 2(s1) + w2 + 2(s2) + w3 + 2(s3)) / 3,
w = time in processor but not being executed, S = # of switches made **before** the process in question completes

= (11 + 16 + 11 + 8 + 15 + 10) / 3

= 71 / 3 = 23.67

Commented [116]: Are we considering any change in process to be a context switch even if it is a CPU burst completing within its allotted time?

Commented [117]: Reschedule should happen every time the timer interrupts. So yes, we need to consider switches even if there is 1 process.

Commented [118]: I am wondering if we should include the very beginning since we need to schedule

Commented [119]: Yes to including it

Commented [120]: Pretty sure they said on piazza

Commented [121]: yea, i saw that on piazza, we do

Commented [122]: Shouldn't it actually be (11 + 2(5)

Commented [123]: What Joe points out is pretty

Commented [124]: I don't think so because while

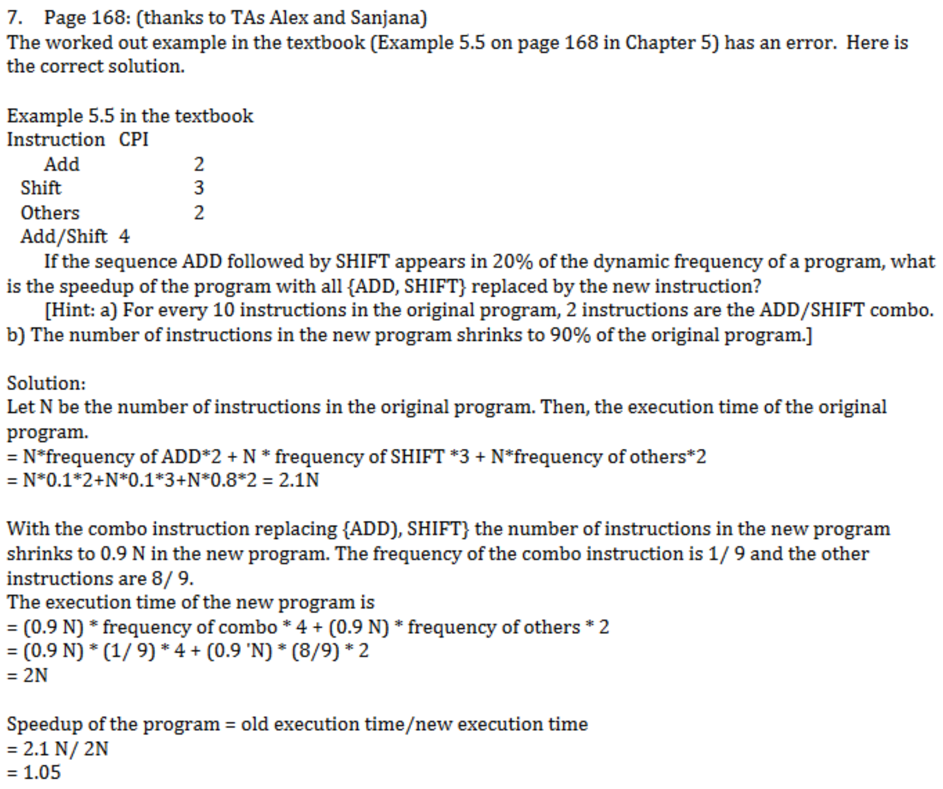
Commented [125]: Oh, I see why. What's on the

Commented [126]: I don't think the current answer

Commented [127]: I find it easier to think about it thi

Commented [128]: agreed! but also the diagram is r

Commented [129]: shouldn't it be 22.3? because like



7. Page 168: (thanks to TAs Alex and Sanjana)

The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:

Let N be the number of instructions in the original program. Then, the execution time of the original program.

= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2

= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.

The execution time of the new program is

= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2

= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2

= 2N

Speedup of the program = old execution time/new execution time

= 2.1 N/ 2N

= 1.05

Commented [130]: There are context switches between same process bursts because the timer will pre-empt after every time slice, so the cost of the context-switch/scheduler is still there.

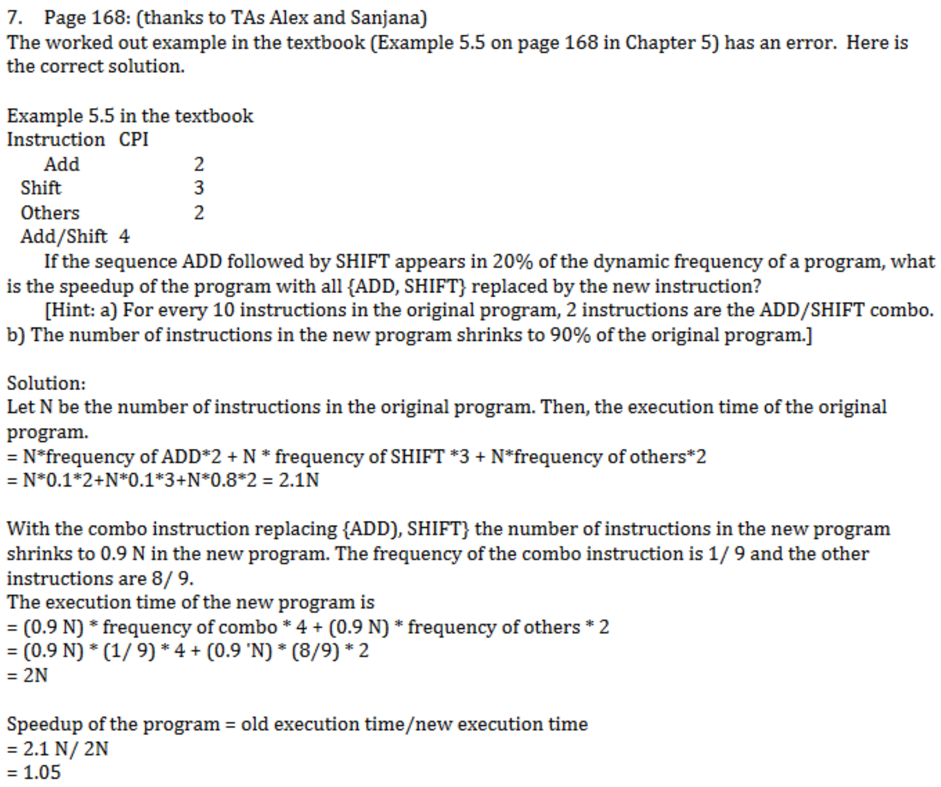
d) Qualitatively, state your intuition on the effect of the choice of timeslice on the waiting time for processes with short CPU bursts. (3 points)

Formatted: Font: Bold, Italic

Better answer (at least based on comments): Based on the results between part A and B, while decreasing the timeslice doesn't seem to change the turnaround/waiting time of long processes (P1), it does shrink the waiting time of processes who have **small CPU bursts**. It allows us to **start every process sooner**, and because P2 and P3 have small CPU bursts, they are able to finish their bursts in at most one timeslice.

e) Qualitatively, state your intuition on the effect of the scheduling and context-switching overhead on the waiting time for processes with short CPU bursts. (3 points)

It seems that for short processes, context switching overhead increases the average wait time. It would be better to just let the CPU run than to interrupt it frequently and make slow progress.



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

For round robin, shorter burst CPU processes take less “rounds” around the scheduling queue then longer burst processes since it’s # of partitions from timeslice will be lower. This means that shorter burst processes won’t be in the scheduling queue for as long and reduces the overall impact of context-switching/scheduling overhead. Although this overhead still does increase the wait time for shorter burst processes, it does not increase it to the extent of longer burst processes that go through more “rounds” in the round robin algorithm.

Question 8. Priority Scheduling (6 Points) (4 minutes)

Assume an operating system with a preemptive priority scheduler that runs two application classes A and B. When processes of type A start, they are assigned a high priority; when processes of type B start, they are assigned a low priority.

7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

a) Describe a scenario in which certain processes in our system could experience starvation. (3 points)

~~Process A would be given CPU resources first because it is a higher priority. If it had a very long CPU burst it could “starve” process B. B would not run until process A had finished.~~

Type A process will be executed before type B process. It will even pause the type B process during its execution. If there is a type B process in the ready queue and type A processes keep arriving and ready, the processor will never execute type B processes until all the type A processes are executed.

b) Describe a policy to prevent processes of type B from starving. (3 points)

After certain intervals, the operating system could increase the priority level of process B so that it eventually attains a higher priority than process A and~~A a~~

Commented [131]: By the same logic proposed, if process B runs first, process A is starved, so it doesn't matter who has the higher priority.

I think the question says about "class A", which can involve many processes and not just one. And the fact that "class A" processes can get serviced first for rounds and rounds before "class B" processes even start should be the main reason.

Commented [132]: I think the order which run first does not matter since it is preemptive. If there is a type B process in the ready queue and type A process keep arrive and ready, type B process will never be executed.

Commented [133]: no my point is that A is a class/type of application, not just one process. the purple text looks good to me

Commented [134]: Oh I see

Commented [135]: This is wrong, this is an example of convoy effect not starvation, the solution below is correct

7. Page 168: (thanks to TAs Alex and Sanjana)

The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:

Let N be the number of instructions in the original program. Then, the execution time of the original program.

= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2

= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.

The execution time of the new program is

= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2

= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2

= 2N

Speedup of the program = old execution time/new execution time

= 2.1 N/ 2N

= 1.05

~~nd~~ has its chance to run on the CPU.~~x~~

Topic Predictions for Question 9, 10???

Can they ask about execution time improvement? They did on the last exam but its still in chapter 5 right?

I think they'll ask us an Instruction Improvement Question.

Busy Bit/Read Pending Process?

Effect of interrupt on the pipeline?

- Finish (drain) anything in execute or later
- Flush anything before execute
- Then run interrupt

Commented [136]: Based on the appendices, I think they do not ask us improvement. They would give us the table for improvement, speed ups, etc.. But do not trust me

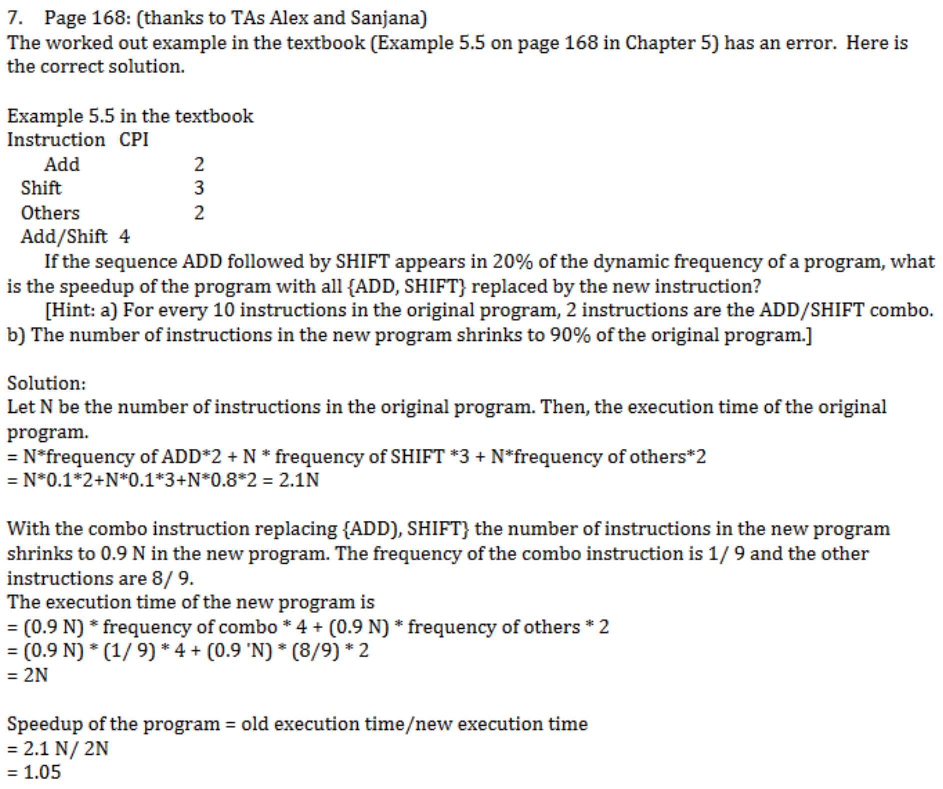
Commented [137]: good point lol

Commented [138]: Dunno about execution time improvement exactly, but I would expect parts of the beginning of Chapter 5 (stuff about CPI and speedups) to be on it.
Also starvation maybe?

Commented [139]: I suppose but starvation is question 8, which we would have just finished, and they usually try to avoid repeating question structures...

Commented [140]: I'm just studying everything just to make sure.

Commented [141]: does this include draining instructions in the fetch stage? or only instructions whose content is in the buffers?



7. Page 168: (thanks to TAs Alex and Sanjana)
The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

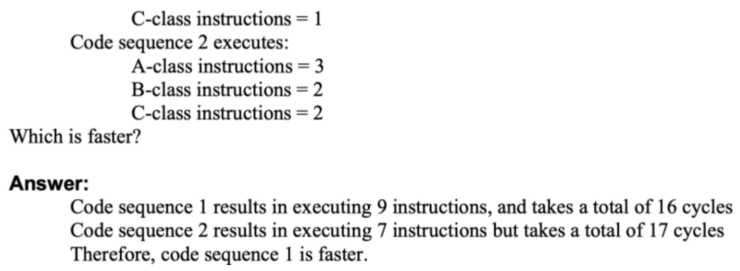
Solution:
Let N be the number of instructions in the original program. Then, the execution time of the original program.
= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.
The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05

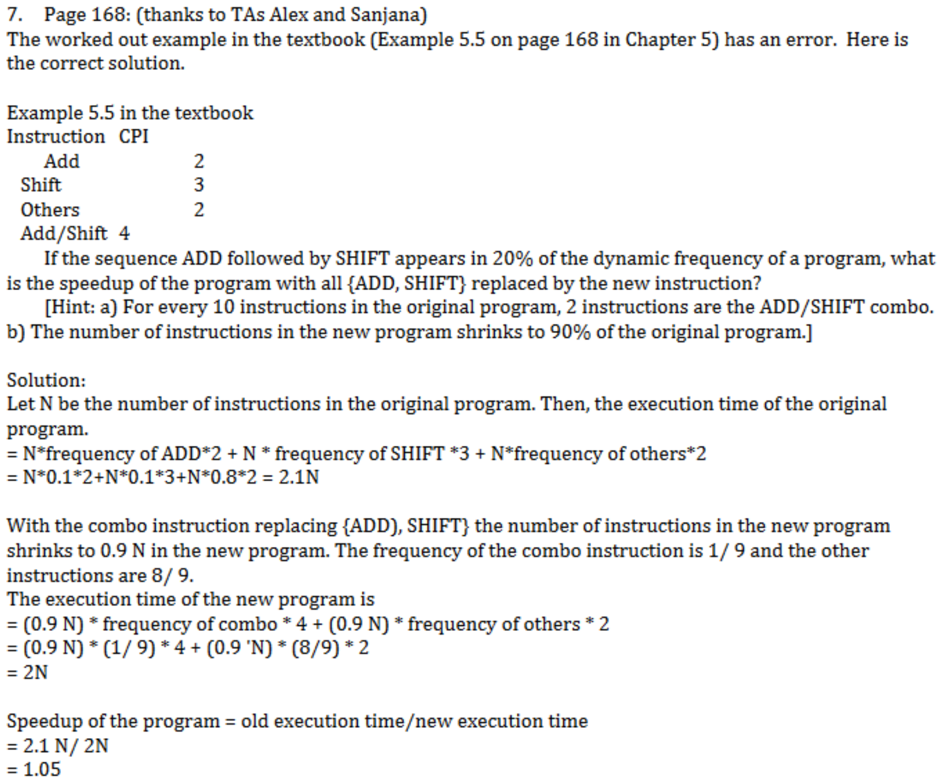
TEXTBOOK EXAMPLES FROM CHAPTERS 5 AND 6:

Put them here
Lecture examples are also important.
Chapter 5:



C-class instructions = 1
Code sequence 2 executes:
A-class instructions = 3
B-class instructions = 2
C-class instructions = 2
Which is faster?

Answer:
Code sequence 1 results in executing 9 instructions, and takes a total of 16 cycles
Code sequence 2 results in executing 7 instructions but takes a total of 17 cycles
Therefore, code sequence 1 is faster.



7. Page 168: (thanks to TAs Alex and Sanjana)

The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:

Let N be the number of instructions in the original program. Then, the execution time of the original program.

= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2

= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.

The execution time of the new program is

= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2

= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2

= 2N

Speedup of the program = old execution time/new execution time

= 2.1 N/ 2N

= 1.05

We might get a question similar to the homeworks but slightly changed like a different scheduling method

Commented [142]: Getting a speedup question asked does not seem unlikely

Commented [143]: Possibly errata in example 5.5?

Commented [144]: very very possible

Commented [145]: or we might get asked the mapping of a different type of priority?

7. Page 168: (thanks to TAs Alex and Sanjana)

The worked out example in the textbook (Example 5.5 on page 168 in Chapter 5) has an error. Here is the correct solution.

Example 5.5 in the textbook

Instruction	CPI
Add	2
Shift	3
Others	2
Add/Shift	4

If the sequence ADD followed by SHIFT appears in 20% of the dynamic frequency of a program, what is the speedup of the program with all {ADD, SHIFT} replaced by the new instruction?

[Hint: a) For every 10 instructions in the original program, 2 instructions are the ADD/SHIFT combo. b) The number of instructions in the new program shrinks to 90% of the original program.]

Solution:

Let N be the number of instructions in the original program. Then, the execution time of the original program.

= N*frequency of ADD*2 + N * frequency of SHIFT *3 + N*frequency of others*2
= N*0.1*2+N*0.1*3+N*0.8*2 = 2.1N

With the combo instruction replacing {ADD), SHIFT} the number of instructions in the new program shrinks to 0.9 N in the new program. The frequency of the combo instruction is 1/ 9 and the other instructions are 8/ 9.

The execution time of the new program is
= (0.9 N) * frequency of combo * 4 + (0.9 N) * frequency of others * 2
= (0.9 N) * (1/ 9) * 4 + (0.9 'N) * (8/9) * 2
= 2N

Speedup of the program = old execution time/new execution time
= 2.1 N/ 2N
= 1.05