Sidharth Gilela
November 2, 2015
CMPE 110

Homework 2

1A) This pipeline diagram would look like this:

	instructions	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
2	xor r0, r0, r0	F	D	X	M	W																										
3	addiu r1, r0, 10		F	D	D	D	X	M	W																							
4	j L1			F	F	F	D	X	M	W																						
5	nop						F	-	-	-	-																					
6	L1: bne r0, r1, -8							F	D	X	M	W																				
7	nop								F	D	-	-	-																			
8	nop									F	-	-	-	-																		
9	loop: lw r3, 0(r2)										F	D	X	M	W																	
10	mul r4, r3, r3											F	D	D	D	X0	X1	X2	X3	M	W											
11	mul r3, r3, r1												F	F	F	D	X0	X1	X2	X3	M	W										
12	addiu r0, r0, 1															F	D	D	D	D	X	M	W									
13	div r3, r4, r3																F	F	F	F	D	D	X0	X0	X0	X0	M	W				
14	sw r3, 0(r2)																				F	F	D	D	D	D	D	D	X	M	W	
15	addiu r2, r2, 4																						F	F	F	F	F	F	D	X	M	W

To calculate the CPI of the entire program I first added the CPI's of the instructions that are not part of the loop. Afterwards, I multiplied the instructions inside the loop by 10 because this program loops ten times. This is seen because R0 has been xor'ed by itself to make it 0 and R1 has the value of 10. The program stops branching when R0 equals R1 and inside the loop, R0 gets incremented by 1 once. Afterwards, I added one more cycle to take care of the last branch instruction that does not loop. After calculating the entire number of cycles, I divided that number by the total number of instructions. Here is how I calculated the CPI of the entire program by using numbers:

The number of cycles:
10 cycles + (10)(21 cycles) = 220 cycles
The number of instructions:
3 instructions + (10)(8 instructions) = 83 instructions
The CPI:
220 cycles/ 83 instructions = 2.65 cycles/instruction

1B) In order to show data bypassing, I have shown this with green arrows. This pipeline diagram would look like this:

	instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
2	xor r0, r0, r0	F	D	X	M	W																			
3	addiu r1, r0, 10		F	D	X	M	W																		
4	j L1			F	D	X	M	W																	
5	nop				F	-	-	-	-																
6	L1: bne r0, r1, -8					F	D	X	M	W															
7	nop						F	D	-	-	-														
8	nop							F	-	-	-	-													
9	loop: lw r3, 0(r2)								F	D	X	M	W												
10	mul r4, r3, r3									F	D	X0	X1	X2	X3	M	W								
11	mul r3, r3, r1										F	D	X0	X1	X2	X3	M	W							
12	addiu r0, r0, 1											F	D	D	D	D	X	M	W						
13	div r3, r4, r3												F	F	F	F	D	X0	X0	X0	X0	M	W		
14	sw r3, 0(r2)																F	D	D	D	D	X	M	W	
15	addiu r2, r2, 4																	F	F	F	F	D	X	M	W

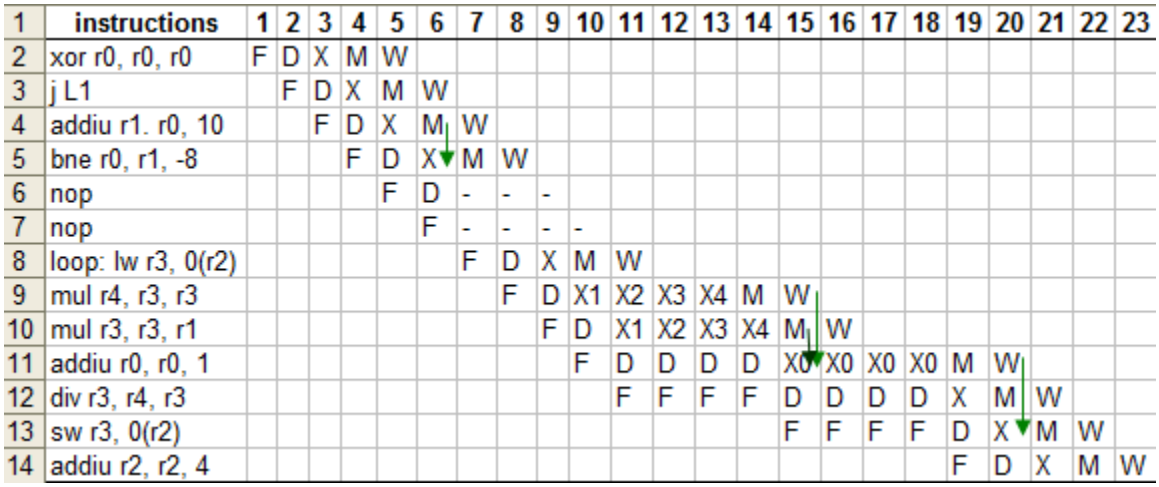
To calculate the CPI of the entire program I first added the CPI's of the instructions that are not part of the loop. Afterwards, I multiplied the instructions inside the loop by 10 because this program loops ten times. This is seen because R0 has been xor'ed by itself to make it 0 and R1 has the value of 10. The program stops branching when R0 equals R1 and inside the loop, R0 gets incremented by 1 once. Afterwards, I added one more cycle to take care of the last branch instruction that does not loop. After calculating the entire number of cycles, I divided that number by the total number of instructions. Here is how I calculated the CPI of the entire program by using numbers:

The number of cycles:
8 cycles + (10)(16 cycles) = 168 cycles
The number of instructions:
3 instructions + (10)(8 instructions) = 83 instructions
The CPI:
168 cycles/ 83 instructions = 2.024 cycles/instruction

1C) The new set of instructions to save one cycle would look like this:

```
1       xor r0, r0, r0
2       j L1
3       addiu r1, r0, 10
4       loop: lw r3, 0(r2)
5       mul r4, r3, r3
6       mul r3, r3, r1
7       addiu r0, r0, 1
8       div r3, r4, r3
9       sw r3, 0(r2)
10      addiu r2, r2, 4
11 L1:  bne r0, r1, -8
```

In order to show bypassing, I have shown this with green arrows. The pipeline diagram would look like this:



1	instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
2	xor r0, r0, r0	F	D	X	M	W																		
3	j L1		F	D	X	M	W																	
4	addiu r1. r0, 10			F	D	X	M	W																
5	bne r0, r1, -8				F	D	X	M	W															
6	nop					F	D	-	-	-														
7	nop						F	-	-	-	-													
8	loop: lw r3, 0(r2)							F	D	X	M	W												
9	mul r4, r3, r3								F	D	X1	X2	X3	X4	M	W								
10	mul r3, r3, r1									F	D	X1	X2	X3	X4	M	W							
11	addiu r0, r0, 1										F	D	D	D	D	X0	X0	X0	X0	M	W			
12	div r3, r4, r3											F	F	F	F	D	D	D	D	X	M	W		
13	sw r3, 0(r2)															F	F	F	F	D	X	M	W	
14	addiu r2, r2, 4																			F	D	X	M	W

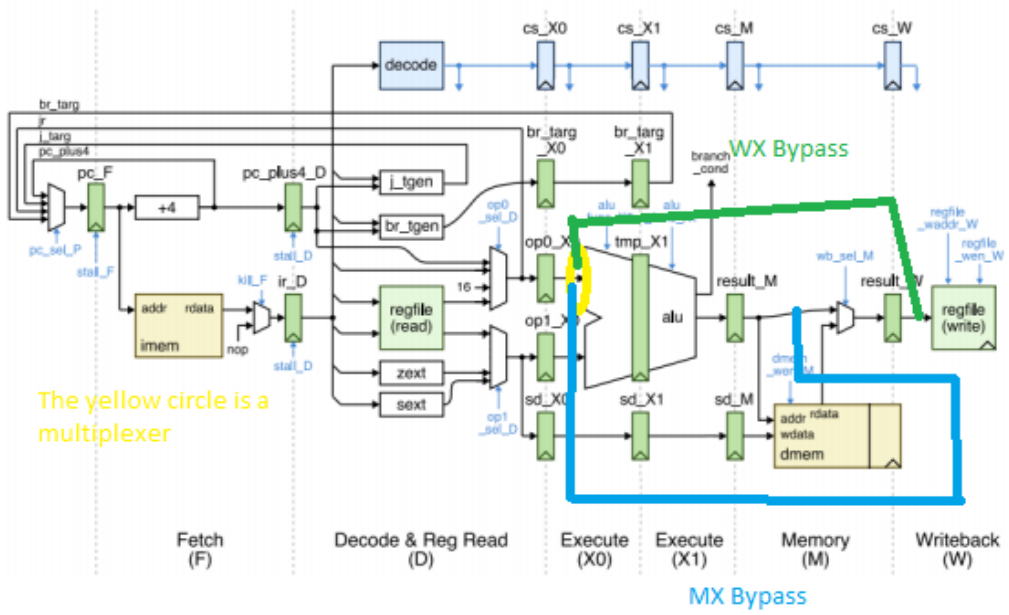
To calculate the CPI of the entire program I first added the CPI's of the instructions that are not part of the loop. Afterwards, I multiplied the instructions inside the loop by 10 because this program loops ten times. This is seen because R0 has been xor'ed by itself to make it 0 and R1 has the value of 10. The program stops branching when R0 equals R1 and inside the loop, R0 gets incremented by 1 once. Afterwards, I added one more cycle to take care of the last branch instruction that does not loop. After calculating the entire number of cycles, I divided that number by the total number of instructions. Here is how I calculated the CPI of the entire program by using numbers:

The number of cycles:
7 cycles + (10)(16 cycles) = 167 cycles
The number of instructions:
3 instructions + (10)(8 instructions) = 83 instructions
The CPI:
167 cycles/ 83 instructions = 2.012 cycles/instruction

2A) I drew the micro architectural RAW dependencies on the pipeline diagram. The RAW dependencies have been shown with blue arrows. When the Register File reads new values in the Registers on the same cycle they are written, I have shown this with green arrows. The pipeline diagram would look like this:

1	instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	bne r1, r0, done	F	D	X0	X1	M	W									
3	lw r5, 0(r2)		F	D	X0	X1	M	W								
4	lw r6, 0(r3)			F	D	X0	X1	M	W							
5	addu r7, r5, r6				F	F	F	D	X0	X1	M	W				
6	addiu r8, r4, 4							F	D	X0	X1	M	W			
7	sw r7, 0(r8)								F	D	D	D	X0	X1	M	W

2B) I have drawn the arrows for an MX bypass and a WX bypass on the diagram below:



---It is not possible to remove stalling due to RAW hazards completely because as seen in my 2C diagram, even with the bypasses the decode instruction in instruction 4 needs to be stalled.

2C) I drew the micro architectural RAW dependencies on the pipeline diagram. In order to show the difference of when bypassing is done, I have show the bypassing color of the arrow as orange. The RAW dependencies have been shown with blue arrows. When the Register File reads new values in the Registers on the same cycle they are written, I have shown this with green arrows. The pipeline diagram would look like this:

1	instructions	1	2	3	4	5	6	7	8	9	10	11	12
2	bne r1, r0, done	F	D	X0	X1	M	W						
3	lw r5, 0(r2)		F	D	X0	X1	M	W					
4	lw r6, 0(r3)			F	D	X0	X1	M	W				
5	addu r7, r5, r6				F	D	D	X0	X1	M	W		
6	addiu r8, r4, 4					F	F	D	X0	X1	M	W	
7	sw r7, 0(r8)							F	D	X0	X1	M	W

3) To calculate the CPI of this question, we have to take into consideration the base CPI, the load proportion, and the branch proportion. Other factors also include the cycle penalties and the percentage of branches taken. The last factor would be the percentage of load instructions that are immediately followed by an instruction that uses the loaded value. I also did 1 minus 0.55 because that is the percentage of how often the branch fails. Here is how to calculate the CPI of this pipeline:

$$1 + (1-0.55)(0.25)(4) + (0.3)(0.12)(2) = 1.522 \text{ cycles per instruction}$$

4A) In order to show when the reservation station is being used, I have shown a single slash looking like this: " / ". In order to show when the reorder buffer is being used, I have shown two slashes looking like this: " // ". The pipeline diagram would look like this:

1	instructions	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
2	xor r0, r0, r0	F	D	X	M	W																								
3	addiu r1, r0, 10		F	D	/	/	X	M	W																					
4	j L1			F	D	X	M	//	//	W																				
5	nop				F	-	-	-	-																					
6	bne r0, r1, -8					F	D	/	/	X	M	W																		
7	nop						F	D	-	-	-																			
8	nop							F	-	-	-	-																		
9	loop: lw r3, 0(r2)								F	D	X	M	W																	
10	mul r4, r3, r3									F	D	/	/	X0	X1	X2	X3	M	W											
11	mul r3, r3, r1										F	D	/	/	X0	X1	X2	X3	M	W										
12	addiu r0, r0, 1											F	D	X	M	//	//	//	//	//	W									
13	div r3, r4, r3												F	D	/	/	/	/	/	/	X0	X0	X0	X0	M	W				
14	sw r3, 0(r2)													F	D	/	/	/	/	/	/	/	/	/	/	/	X	M	W	
15	addiu r2, r2, 4														F	D	X	M	//	//	//	//	//	//	//	//	//	//	//	W

4B) In order to show when the reservation station is being used, I have shown a single slash looking like this: “ / ”. In order to show when the reorder buffer is being used, I have shown two slashes looking like this: “ // ”. In order to show when bypassing is being done, I have shown this with a green arrow. The pipeline diagram would look like this:

1	instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
2	xor r0, r0, r0	F	D	X	M	W																		
3	addiu r1, r0, 10		F	D	X	M	W																	
4	j L1			F	D	X	M	W																
5	nop				F	-	-	-	-															
6	bne r0, r1, -8					F	D	X	M	W														
7	nop						F	D	-	-	-													
8	nop							F	-	-	-	-												
9	loop: lw r3, 0(r2)								F	D	X	M	W											
10	mul r4, r3, r3									F	D	X1	X2	X3	X4	M	W							
11	mul r3, r3, r1										F	D	X1	X2	X3	X4	M	W						
12	addiu r1, r0, 10											F	D	X	M	//	//	//	W					
13	div r3, r4, r3												F	D	//	//	X0	X0	X0	X0	M	W		
14	sw r3, 0(r2)													F	D	/	/	/	/	/	X	M	W	
15	addiu r2, r2, 4														F	D	X	M	//	//	//	//	//	W

4C) For 4A in order to calculate the CPI, I first added the CPI’s of the instructions that are not part of the loop. Afterwards, I multiplied the instructions inside the loop by 10 because this program loops ten times. This is seen because R0 has been xor’ed by itself to make it 0 and R1 has the value of 10. The program stops branching when R0 equals R1 and inside the loop, R0 gets incremented by 1 once. Afterwards, I added one more cycle to take care of the last branch instruction that does not loop. After calculating the entire number of cycles, I divided that number by the total number of instructions. Here is how I calculated the CPI of the entire program by using numbers:

The number of cycles:
9 cycles + (1)(20 cycles) + (9)(19 cycles) = 200 cycles
--- The reason why I did 20 cycles for the first time and 19 cycles for the rest of the 9 times is because the first time the branch is done is takes 2 cycles. However, for the rest of the 9 times, it does only one cycle.
The number of instructions:
3 instructions + (10)(8 instructions) = 83 instructions
The CPI:
200 cycles/ 83 instructions = 2.410 cycles/instruction

For 4B in order to calculate the CPI, I first added the CPI’s of the instructions that are not part of the loop. Afterwards, I multiplied the instructions inside the loop by 10 because this program loops ten times. This is seen because R0 has been xor’ed by itself to make it 0 and R1 has the value of 10. The program stops branching when R0 equals R1 and inside the loop, R0 gets incremented by 1 once. Afterwards, I added one more cycle to take care of the last branch instruction that does not loop. After calculating the entire

number of cycles, I divided that number by the total number of instructions. Here is how I calculated the CPI of the entire program by using numbers:

The number of cycles:
8 cycles + (10)(15 cycles) = 158 cycles
The number of instructions:
3 instructions + (10)(8 instructions) = 83 instructions
The CPI:
158 cycles/ 83 instructions = 1.904 cycles/instruction