**CMPE 110 Computer Architecture Winter 2016, Homework #2**

### Computer Engineering UC Santa Cruz (Update 1)

January 27, 2016

**Name:**  Gurpreet Singh

#### Email: gsingh13@ucsc.edu

**Submission Guidelines:**

* This homework is due on Sunday 02/09/16.
* The homework must be submitted to ecommons by 11:59pm.
  + Anything later is a late submission and will be subject to penalties, as per the late submission policy.

#### Please write your name and your UCSC email address .

* **The homework should be “readable” without too much effort**
  + The homework must be typed and submitted as a single file in PDF format
  + Please name your homework file **cmpe110-hw2-yourcruzid.pdf**
  + Please keep your responses coherent and organized or you may lose points
* Provide details on how to reach a solution. **An answer without explanation gets no credit. Clearly state all assumptions.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question** | **Part A** | **Part B** | **Part C** | **Total** |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |
| **4** |  |  |  |  |
| **Total** |  |  |  |  |

*•* Points: 78 = 18 + 18 + 21 + 21

# Question 1. Pipeline Stages (18 points)

Consider a five stage pipelined (fetch, decode, execute, memory, write back) processor.

|  |  |
| --- | --- |
| **Pipline Stage** | **Latency of each stage (ps)** |
| Fetch (F0, F1) | 240 |
| Decode (D) | 320 |
| Execute (X0, X1, X2) | 280 |
| Memory (M) | 400 |
| Write Back (W) | 200 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Processor** | **Cycle**  **Time** | **Max Clock**  **Frequency** | **Latency of**  **Instruction** | **Throughput** |
| a) Baseline | 850 ns | .0025 hz | 6800 sec/inst | .001176 |
| b) Faster memory | 850 ns | .003125 hz | 6800 sec/inst | .001176 |
| c) Proposed scheme | 850 ns | .002 hz | 6800 sec/inst | .001176 |

## Question 1.A Baseline (6 points)

Fill out the first row in the table above for the given pipelined processor. Cycle time is the clock period at which this machine can run. Max clock frequency is the maximum frequency clock that can be run given the cycle time (lower frequencies are possible). Latency of instruction is the time it takes to get the first output after it starts. Throughput is the rate at which output is produced.

## Question 1.B Faster Memory (6 points)

Suppose there is an optimization that reduces the Memory stage latency by 100 ps. However, the optimization in memory stage results in 100ps increase in the Write back stage latency. How much would this improve the overall performance of the 8-stage pipeline? Fill out the second row in the table above for this new pipeline.

## Question 1.C Proposed Scheme (6 points)

If you had the option to rearrange the pipeline, how would your arrange the pipeline to achieve maximum frequency?

For question 1,

* F0, F1 – 240 each so 240 + 240 = 480 total
* X1, X2, X3 = 280 + 280 + 280 = 840 total
* Here, we know that pipelining to 5 stages reduces the cycle time cycle and needs to be longer in order to accommodate the register at the end of the stage.
* The cycle time of the slowest stage is all the X stage = 840 and extra time needed for the register and therefore we have 850 ns.
* Latency = 8 stages (cycle time) = 8\*850 = 6800
* Throughput = 1/cycle time = 1/850 = .001176
* Max Clock Frequency = Baseline = 1/400 = .0025, Faster Mem = 1/320 = .003125, proposed scheme = 1/500 = .002

# Question 2. Datapath Bypassing (18 points)

In this problem we will be investigating the implications of using a two-cycle pipelined integer ALU. Our new pipelined processor will have the following six stages:

* F - instruction fetch
* D - decode and read registers
* X0 - first half of the ALU operation
* X1 - second half of the ALU operation
* M - data memory read/write
* W - write registers

The figure on the next page shows the datapath of the new six-stage datapath. Notice that this datapath currently does not allow any data bypassing. Also notice that conditional branches are resolved by the end of the X1 stage. Assume that this instruction set does not include a branch delay slot.

1. add r3, r1, r2
2. xor r7, r3, r4
3. and r5, r2, r3
4. xor r8, r8, r8
5. add r8, r7, r3

6 sw r5, 400(r3)

## Question 2.A Implementing Bypassing (6 points)

Modify the figure on the next page to implement the different types of bypassing.

## Question 2.B Stalling (6 points)

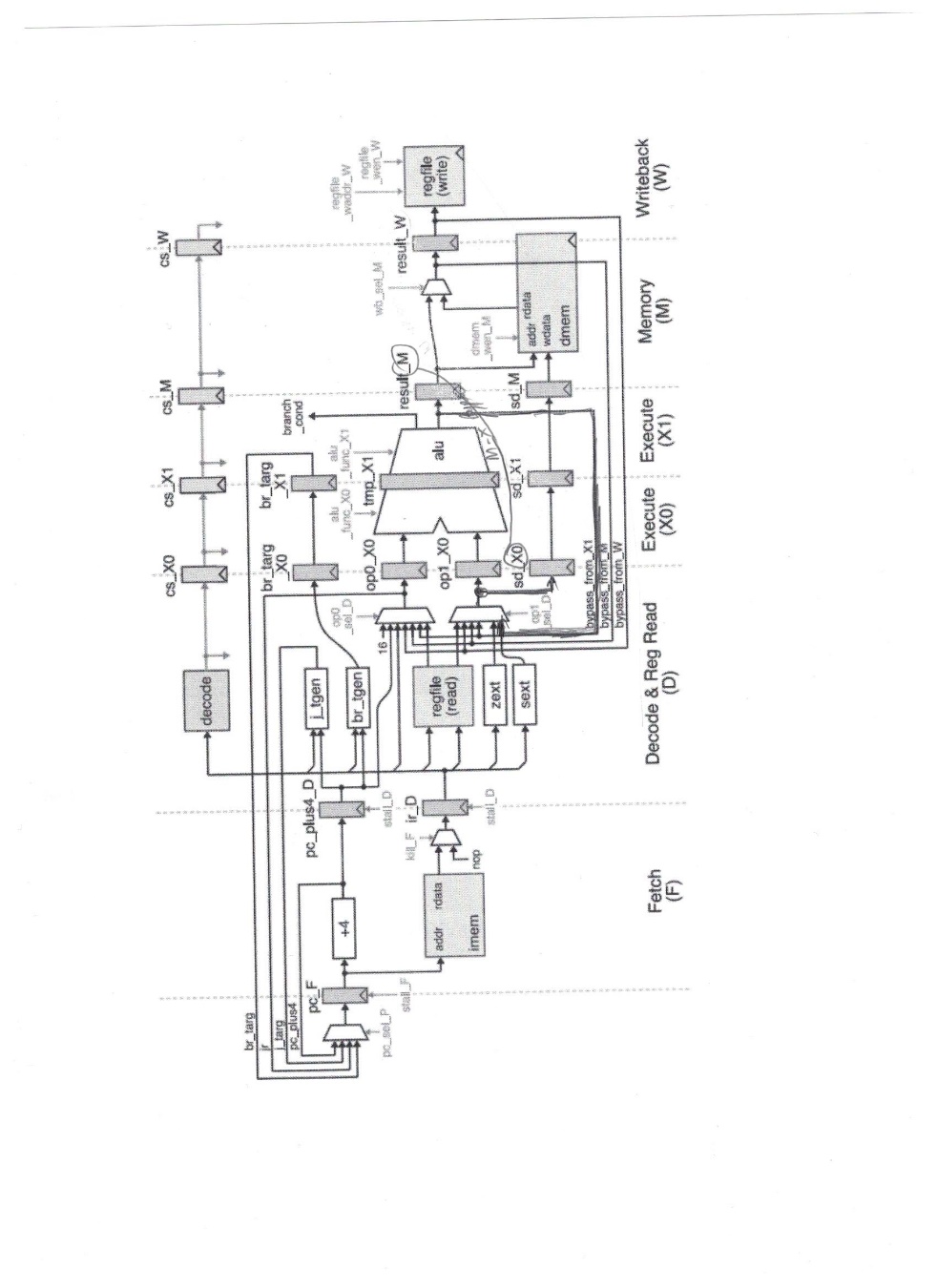
For the above assembly code, draw a pipeline diagram using stalls to resolve any data de- pendencies that you find.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instructions | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Add r3, r1, r2 | F | D | X0 | X1 | M | W |  |  |  |  |  |  |  |  |  |  |
| Xor r7, r3, r4 |  | F | D | X0 | X1 | M | W |  |  |  |  |  |  |  |  |  |
| And r5, r2, r3 |  |  | F | D | X0 | X1 | M | W |  |  |  |  |  |  |  |  |
| Xor r8, r8, r8 |  |  |  | F | D | D | D | X0 | X1 | M | W |  |  |  |  |  |
| Add r8, r7, r3 |  |  |  |  | F | F | F | D | X0 | X1 | M | W |  |  |  |  |
| Sw r5, 400(r3) |  |  |  |  |  |  |  | F | D | D | D | D | Xo | X1 | M | W |

## Question 2.C Bypassing (6 points)

For the above assembly code, draw a pipeline diagram using whichever bypass you think is valid for that case. Indicate which type of bypass you have used for each case.

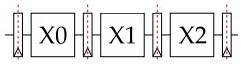
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instructions | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Add r3, r1, r2 | F | D | X0 | X1 | M | W |  |  |  |  |  |  |  |  |  |  |
| Xor r7, r3, r4 |  | F | D | X0 | X1 | M | W |  |  |  |  |  |  |  |  |  |
| And r5, r2, r3 |  |  | F | D | X0 | X1 | M | W |  |  |  |  |  |  |  |  |
| Xor r8, r8, r8 |  |  |  | F | D | D | D | X0 | X1 | M | W |  |  |  |  |  |
| Add r8, r7, r3 |  |  |  |  | F | F | F | D | X0 | X1 | M | W |  |  |  |  |
| Sw r5, 400(r3) |  |  |  |  |  |  |  | F | D | D | Xo | X1 | M | W |  |  |



# Question 3. Pipelining and Hazzards (21 points)

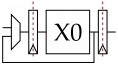
Consider the MIPS assembly code given below. We want to run this code on a 5-stage pipelined processor, with some modifications. The processor is a typical 5-stage pipeline (F-D-X-M-W), with the following exceptions:

* + The multiplier block used to execute the mul instruction is pipelined into three stages as shown:



This means that a multiply instruction runs through the pipeline as follows: F-D-X0- X1-X2-M-W and **up to three multiply instructions maybe in-flight at a time. All other instruction types are blocked from the execute stage while any of the multiply stages are being used.**

* + The divider block used to execute the div instruction is iterative and takes five cycles as shown:



This means that a divide instruction runs through the pipeline as follows: F-D-X0- X0-X0-X0-X0-M-W. **All other instructions are blocked from the execute stage while a division is being done.**

1. xor r0, r1, r1
2. addiu r1, r0, 16
3. j L1
4. loop: lw r3, 0(r2)
5. mul r4, r3, r3
6. div r3, r4, r3
7. mul r3, r3, r1
8. mul r3, r3, r0
9. addiu r0, r0, 2

10 sw r3, 0(r2)

11 addiu r2, r2, 4

12 L1: bne r0, r1, -9

## Question 3.A Structural Hazards (7 points)

Draw a pipeline diagram showing the execution of the MIPS code through the first iteration of the loop, without bypassing. **Assume data hazards and structural hazards are resolved using only stalling.** Assume the processor assumes branches are not taken, until they are resolved. What is the CPI of the entire program?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| Xor r0, r1, r1 | F | D | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| addi r1, r0, r6 |  | F | D | D | D | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| j Li |  |  | F | F | F | D | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Loop: lw r3, r0 |  |  |  |  |  | F | - | - | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| mul r4, r3, r3 |  |  |  |  |  |  | F | D | X0 | X1 | X2 | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |
| div r3, r4, r3 |  |  |  |  |  |  |  | F | D | D | D | X0 | X0 | X0 | X0 | X0 | M | w |  |  |  |  |  |  |  |  |
| mul r3, r3, r1 |  |  |  |  |  |  |  |  | F | F | F | D | D | D | D | D | X0 | X1 | X2 | M | W |  |  |  |  |  |
| mul r3, r3, r0 |  |  |  |  |  |  |  |  |  |  |  | F | F | F | F | F | D | X0 | X1 | X2 | M | W |  |  |  |  |
| addiv r0, r0, 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | D | D | X | M | W |  |  |  |
| sw r3, 0(r2) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | F | F | D | X | M | W |  |  |
| addiu r2, r2,4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | X | M | W |  |
| Li: bne r0, r1, -9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | X | M | W |

* Prior to loop:10 cycles, loop = 21 cycles, 10 iterations = 210 cycle, loop ends = 1
* Total cycles :10 + 210 + 1 = 221 cycles
* Total instruction: 3 + 10 \* 9 + 1 = 94 inst
* CPI : 221/94 = 2.351 cycles/ inst

## Question 3.B Data Hazards (7 points)

Draw a pipeline diagram similar to Part A, but now **assume the processor has data bypassing.** What is the CPI of the entire program?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| Xor r0, r1, r1 | F | D | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| addi r1, r0, r6 |  | F | D | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| j Li |  |  | F | D | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Loop: lw r3, r0 |  |  |  | F | - | - | - | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| mul r4, r3, r3 |  |  |  |  | F | D | X0 | X1 | X2 | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| div r3, r4, r3 |  |  |  |  |  | F | D | D | D | X0 | X0 | X0 | X0 | X0 | M | W |  |  |  |  |  |  |  |  |  |  |
| mul r3, r3, r1 |  |  |  |  |  |  | F | F | F | D | D | D | D | D | X0 | X1 | X2 | M | W |  |  |  |  |  |  |  |
| mul r3, r3, r0 |  |  |  |  |  |  |  |  |  | F | F | F | F | F | D | X0 | X1 | X2 | M | W | F |  |  |  |  |  |
| addiv r0, r0, 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | D | D | X | M | W |  |  |  |  |  |
| sw r3, 0(r2) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | F | F | D | X | M | W | F |  |  |  |
| addiu r2, r2,4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | X | M | W | F |  |  |
| Li: bne r0, r1, -9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | X | M | W | F |  |

* Prior to loop: 8 cycles, loop = 17 cycles, 10 iterations = 170 cycle, loop ends = 1
* Total cycles: 8 + 170 + 1 = 179 cycles
* Total instruction: 3 + 10 \* 9 + 1 = 94 inst
* CPI: 179/94 = 1.904 cycles/ inst

## Question 3.C Control Hazards(7 points)

Will the assembly code shown above lead to control hazards? If yes, Where does it occur and how can it be solved?

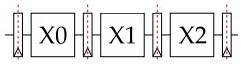
* The control hazard happens when the loop starts. This can be solved by pipeline stall cycles, where the pipeline is frozen until the branch outcome and target and known. Following this, we can proceed with the fetch.

# Question 4. Out-of-Order Execution (21 points)

#### This question is based on the same pipeline and code as question 3.

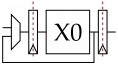
Consider the MIPS assembly code given below. We want to run this code on a 5-stage pipelined processor, with some modifications. The processor is a typical 5-stage pipeline (F-D-X-M-W), with the following exceptions:

* The multiplier block used to execute the mul instruction is pipelined into three stages as shown:



This means that a multiply instruction runs through the pipeline as follows: F-D-X0- X1-X2-M-W and **up to three multiply instructions maybe in-flight at a time. All other instruction types are blocked from the execute stage while any of the multiply stages are being used.**

* The divider block used to execute the div instruction is iterative and takes five cycles as shown:



This means that a divide instruction runs through the pipeline as follows: F-D-X0- X0-X0-X0-X0-M-W. **All other instructions are blocked from the execute stage while a division is being done.**

#### This processor supports out of order execution.

1. xor r0, r1, r1
2. addiu r1, r0, 16
3. j L1
4. loop: lw r3, 0(r2)
5. mul r4, r3, r3
6. div r3, r4, r3
7. mul r3, r3, r1
8. mul r3, r3, r0
9. addiu r0, r0, 2

10 sw r3, 0(r2)

11 addiu r2, r2, 4

12 L1: bne r0, r1, -9

**Question 4.A Out-of-Order with no Bypassing (7 points)**

Draw a pipeline diagram **showing the out-of-order execution** of the MIPS code through the first iteration of the loop, without bypassing. **Assume data hazards and structural hazards are resolved using only stalling.** Assume the processor assumes branches are not taken, until they are resolved.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| Xor r0, r1, r1 | F | D | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| addi r1, r0, r6 |  | F | D | R2 | R3 | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| j Li |  |  | F | D | X | M | R0 | R0 | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Loop: lw r3, r0 |  |  |  |  |  | F | - | - | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| mul r4, r3, r3 |  |  |  |  |  |  | F | D | X0 | X1 | X2 | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |
| div r3, r4, r3 |  |  |  |  |  |  |  | F | D | R0 | R0 | X0 | X0 | X0 | X0 | X0 | M | W |  |  |  |  |  |  |  |  |
| mul r3, r3, r1 |  |  |  |  |  |  |  |  | F | D | R0 | R0 | R0 | R0 | R0 | R0 | X0 | X1 | X2 | M | W |  |  |  |  |  |
| mul r3, r3, r0 |  |  |  |  |  |  |  |  |  |  |  | F | D | R0 | R0 | R0 | D | X0 | X1 | X2 | M | W |  |  |  |  |
| addiv r0, r0, 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | R0 | R0 | X | M | W |  |  |  |
| sw r3, 0(r2) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | R0 | R0 | X | M | W |  |  |
| addiu r2, r2,4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | X | M | W |  |
| Li: bne r0, r1, -9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | X | M | W |

## Question 4.B Out-of-Order with Bypassing (7 points)

Draw a pipeline diagram similar to Part A, showing the out-of-order execution, but now

#### assume the processor has data bypassing.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| Xor r0, r1, r1 | F | D | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| addi r1, r0, r6 |  | F | D | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| j Li |  |  | F | D | X | M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Loop: lw r3, r0 |  |  |  |  |  | F | - | - | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| mul r4, r3, r3 |  |  |  |  |  |  | F | D | X0 | X1 | X2 | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |
| div r3, r4, r3 |  |  |  |  |  |  |  | F | D | R0 | R0 | X0 | X0 | X0 | X0 | X0 | M | W |  |  |  |  |  |  |  |  |
| mul r3, r3, r1 |  |  |  |  |  |  |  |  | F | D | R0 | R0 | R0 | R0 | R0 | R0 | X0 | X1 | X2 | M | W |  |  |  |  |  |
| mul r3, r3, r0 |  |  |  |  |  |  |  |  |  |  |  | F | D | R0 | R0 | R0 | D | X0 | X1 | X2 | M | W |  |  |  |  |
| addiv r0, r0, 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | R0 | R0 | X | M | W |  |  |  |
| sw r3, 0(r2) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | R0 | R0 | X | M | W |  |  |
| addiu r2, r2,4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | X | M | W |  |
| Li: bne r0, r1, -9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | X | M | W |

**Question 4.C Program Latency (7 points)**

Determine the number of cycles it takes to execute all iterations of the loop for both the scenario in Part A and the scenario in Part B. Justify your answer.

With No Bypassing:

* First iteration of loop starts at Cycle 4, ends on 30 so total of 27 cycles needed in the first iteration.
* BNE fetches 2 instructions but these 2 instructions do not count toward the cycle so BNE from second only cost 1 cycle, after the branch is resolved the remainder of the second iteration onward behave like 13- 30 cycle which is 18 cycles.
* Each iteration after the first adds 19 cycles to total cycle count.
* Total = 27 + 9 \* 19 = 198 cycles.

With Bypassing

* First iteration between 3-23 cycle = 20 cycles. 2 BNE towards 1 cycle.
* Branch is resolved 11-23 only 23 cycles.
* Each iteration adds 14 cycles.
* Total = 20 + 9 \* 14 = 146 cycles.