CS/CoE 1541 – Exam 1 (Spring 2019). Name: ______________________________

Question 1 (8+2+2+3=15 points): In this problem, consider the execution of the following code segment on a 5-stage pipeline with forwarding/stalling hardware and branches resolved in ID. Assume that the loop I2-I5 in the following code executes 1000 times (branches are not delayed branches).

```
I1: sub    $3, $0, $0
I2: lw     $1, 0($2)
I3: add    $3, $3, $1
I4: addi   $2, $2, 4
I5: bneq   $5, $2, I2      /*branch to I2 if contents of $5 and $2 are not equal*/
I6: sw     $3, 0($2)
```

(a) Complete the following trace for the execution of the above code (up to cycle 15) assuming no branch prediction (that is always predicting that a branch is not taken).

	IF	ID	EX	MEM	WB
Cycle 1	sub				
Cycle 2	lw	sub			
Cycle 3	add	**lw**	sub		
Cycle 4	addi	add	**lw**	sub	
Cycle 5	addi	add	🟧	**lw**	sub
Cycle 6	bneq	addi	add	🟧	**lw**
Cycle 7	sw	bneq	addi	add	🟧
Cycle 8	lw		bneq	addi	add
Cycle 9	add	**lw**		bneq	addi
Cycle 10	add		lw		bneq
Cycle 11	addi	add		lw	
Cycle 12	bneq	addi	add		**lw**
Cycle 13	sw	bneq	addi	add	
Cycle 14	lw		bneq	addi	add

(b) How many cycles would it take to complete the execution of the 1000 iterations of the loop (4000 instructions) – still assuming no branch prediction?

6000

(c) What would be the CPI while executing the loop (with no branch prediction)?

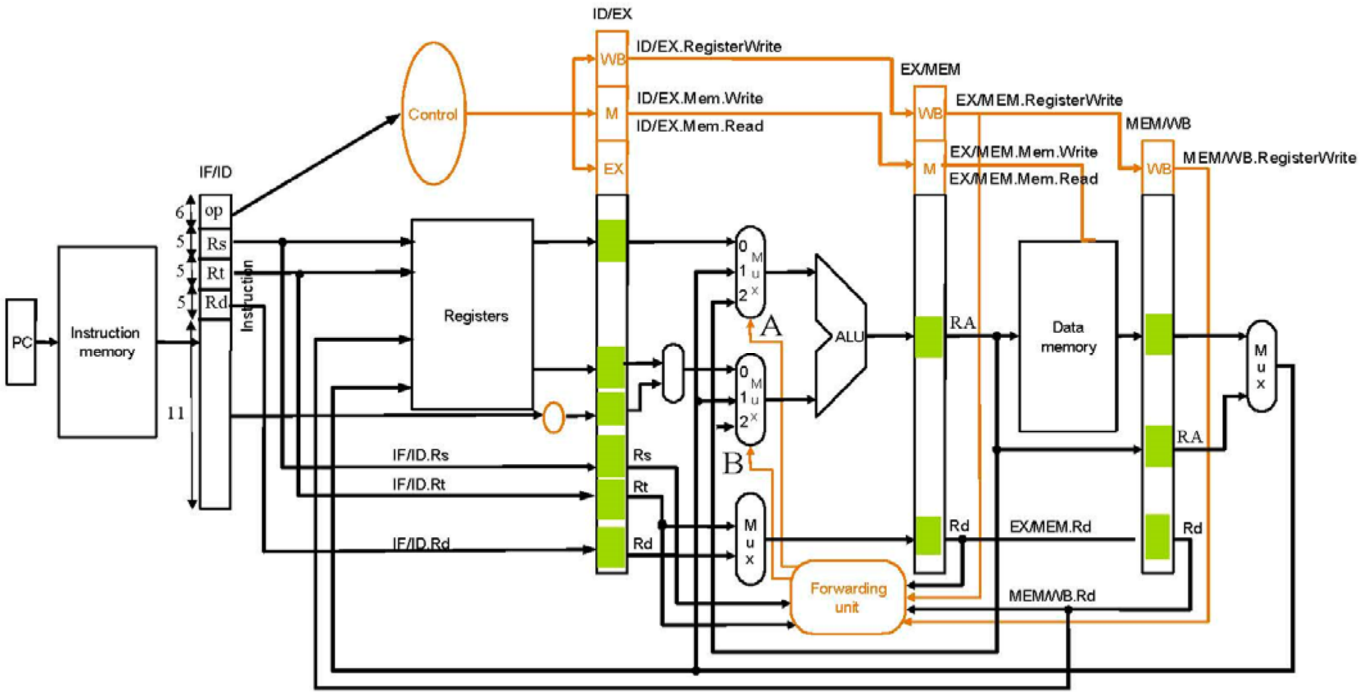
6000/4000 = 1.5

(d) What would be the CPI while executing the loop is we assume a perfect branch predictor?

5000/4000 = 1.25

Question 2 (7+5+4+4=20 points): Consider the execution of the following code on the shown five stage pipeline and assume that at the beginning of cycle 1, the PC contains the address FA00:

```
FA00:  add   $3, $2, $1        /* See last page of exam bulletin for */
FA04:  lw    $5, $4(120)       /* MIPS machine instruction formats */
FA08:  sw    $1, $3(120)
FA0B:  sub   $6, $7, $8
FA10:  beq   $2, $6, FA80
```



a) Specify the values of the following pipeline buffer registers at the beginning of cycle 5 (when the add instruction is in MEM/WB. Use X if the value cannot be determined and is inconsequential.

PC: FA10	IF/ID.Rs: 7	ID/EX.Rs: 3	EX/MEM.Rd: 5	MEM/WB.Rd: 3
	IF/ID.Rt: 8	ID/EX.Rt: 1		
	IF/ID.Rd: 6	ID/EX.Rd: X		

b) Specify the values of the following control signals at the beginning of cycle 5:

ID/EX.RegWrite: 0	EX/MEM.RegWrite: 1	MEM/WB.RegWrite: 1
ID/EX.MemWrite: 1	EX/MEM.MemWrite: 0	
ID/EX.MemRead: 0	EX/MEM.MemRead: 1	

c) Specify the values of the signals controlling the forwarding muxes at the beginning of cycle 5:

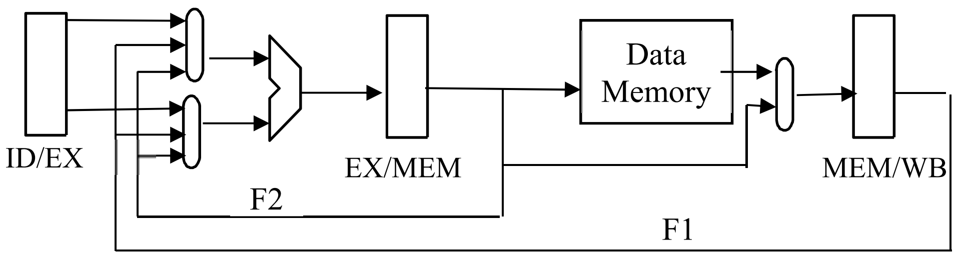
Signal A: 1 Signal B: 0

d) Assuming that at cycle 1, registers 1, 2, 3 and 4 contain the decimal values 100, 200, 300 and 400, respectively, specify (in decimal) the content of the following registers at the beginning of cycle 5 (again use X if the value cannot be determine)?

EX/MEM.RA: 400+120 = 520 MEM/WB.RA: 100+200=300

Question 3 (2.5*6=15 points): Consider a 5-stage pipeline with the following forwarding paths:

- F1 between the MEM/WB inter-stage buffer and the input of the EX stage and,
- F2 between the EX/MEM inter-stage buffer and the input of the EX stage.



(a) For each of the following five code segments, identify which of the statements (i), (ii), (iii), (iv) or (v) applies? Indicate your answer by circling the correct answer for each segment.
   (i) The data hazard is completely resolved by F1
   (ii) The data hazard is completely resolved by F2
   (iii) The data hazard is resolved by stalling the pipeline and then using F1
   (iv) The data hazard is resolved by stalling the pipeline and then using F2
   (v) There is no data hazard.

Segment 1: add r1, r2, r3
sub r4, r1, r2
beq r2, r3, L
(i) [✓ (ii)] (iii) (iv) (v)

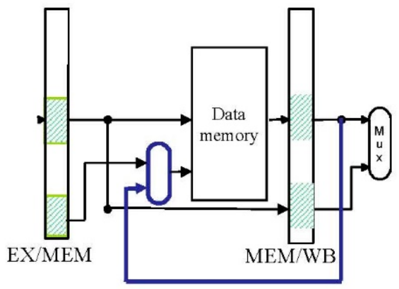
Segment 2: lw r1, 300(r2)
sub r4, r1, r2
add r5, r1, r2
(i) (ii) [✓ (iii)] (iv) (v)

Segment 3: sw r2, 304(r1)
add r1, r2, r3
sw r2, 300(r3)
(i) (ii) (iii) (iv) [✓ (v)]

Segment 4: add r1, r2, r3
lw r3, r6(100)
sub r4, r1, r2
[✓ (i)] (ii) (iii) (iv) (v)

Segment 5: lw r1, 300(r2)
add r2, r6, r7
sub r4, r1, r3
[✓ (i)] (ii) (iii) (iv) (v)

(b) Give an example of a short code segment (2 or 3 instructions similar to the ones in part a) which can utilize a forwarding path from the MEM/WB inter-stage buffer to the input of the MEM stage (shown in the figure).



lw r1, r2(100)
sw r1, r2(100)

Question 4 (5*3 = 15 points): Consider a pipelined CPU with **6-stages** (IF, ID, EX1, EX2, MEM, WB) with branch conditions and target addresses being resolved in the EX2 stage. The delays for the six stages are 1.7, 1.9, 1.8, 2.0, 1.7 and 2.0 ns respectively. Assume that, on average, 30% of the instructions executed on this architecture are load/store instructions and 20% are branch instructions with 30% of the executed branches taken. The CPI of the architecture is 3.0 if the stalls due to control hazards are ignored.

(a) What is the CPI for the architecture when stalls due to control hazards are included?

- CPI_a = 3.0 + 0.2 * 0.3 * 3 = 3.18

(b) Repeat part (a) if the architecture is modified such that the logic for calculating the branch target and address is moved from EX2 to ID. This modification, however, will increase the delay of the ID stage from 1.9 to 2.1 ns and reduce the delay of the EX2 stage from 2.0 to 1.8 ns.

- CPI_b = 3.0 + 0.2 * 0.3 * 1 = 3.06

(c) Repeat part (a) if branches are resolved in EX2 but hardware for branch prediction is added to IF resulting in a 90% prediction accuracy. That is, after a branch, the correct instruction is fetched 90% of the time and the wrong instruction 10% of the time. The branch predictor will increase the delay of the IF stage from 1.7 to 1.9 nanoseconds.

- CPI_c = 3.0 + 0.2 * 0.1 * 3 = 3.06

(d) What is the cycle time for each of the above three architectures?

- T_a = the cycle time for the original architecture = 2.0

- T_b = the cycle time when branch resolution is moved to ID as described in (b) = 2.1

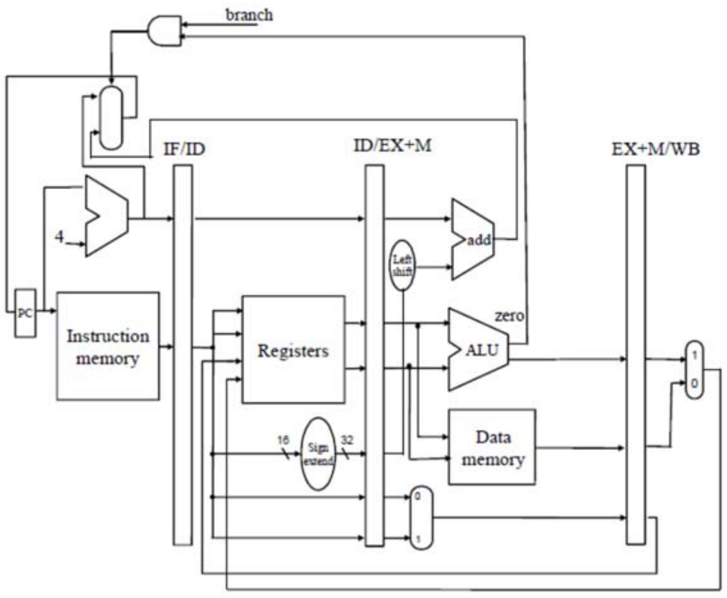
- T_c = the cycle time when a branch predictor is added as described in (c) = 2.0

(e) How would you determine which of the three architectures is the most efficient?

The one with the smallest value of CPI * T is the most efficient
In this case, it is the architecture of part c

Question 5 (5*3 = 15 points): In this question, you will explore changing the MIPS ISA so that lw/sw instructions do not use an immediate constant. That is, the memory address is found in the register without the capability of adding a constant (this is called register indirect addressing). With this modification, it is possible to have a 4-stage pipeline by merging the EX and MEM stages into one stage (call it EX+M), as shown in the figure below.

The advantage of the 4-stage pipeline is that forwarding completely eliminates data hazards. Its disadvantage is that the **number of executed instructions in a program increases by 10%** because each instruction of the form *"lw $r1, I($r2)"* has to be replaced by two instructions (*"addi $r2, $r2, I"* followed by *"lw $r1, $r2"*) and each instruction of the form *"sw $r1, I($r2)"* has to be replaced by two instructions (*"addi $r2, $r2, I"* followed by *"sw $r1, $r2"*).



(a) Assume that it takes 90ps to read from the register file, 90ps to write into the register file and 200ps to fetch or store into memory (instruction or data). Assume also that the Adder and ALU delays are 150ps and 240ps, respectively and ignore all other delays. What is the minimum cycle time for this 4-stage pipeline?

240

(b) To compare the performance of the 4-stage pipeline with the original 5-stage pipeline, we assume that both have the same cycle time, use forwarding and resolve branches in the EX stage.

a. Given that the CPI for the 5-stage pipeline is 2.05 and that the probability of a load-use data hazard in the 5-stage pipeline is 5% (the probability that an instruction is a *lw* instruction that loads into a register R and that this *lw* instruction is immediately followed by an instruction which reads from R). Compute the CPI of the 4-stage pipeline.

$CPI_{5\text{-stage}} = 2.05$

$CPI_{4\text{-stage}} = 2.05 - 0.05 * 1 = 2$

(c) Which pipeline would be more efficient? – should quantitatively support your answer

Clock cycles to complete for 5 stage = CPI * # of instructions = 2.05 X
Clock cycles to complete for 4 stage = CPI * # of instructions = 2.0 * 1.1X =2.2X

Question 6 (8+4+3=15 points): Consider the scheduling of the following loop on a superscalar architecture with two pipelines, one for ALU/branch instructions and the other for lw/sw instructions.

```
L1:   lw     $t0, 1000($s4)
L2:   addi   $s1, $s1, 1
L3:   lw     $t1, 1004($s4)
L4:   addi   $s5, $t1, 4
L5:   add    $t0, $t0, $s1
L6:   sw     $s1, 5000($s4)
L7:   beq    $s4, $s6, L
```

a) Show the scheduling of one iteration of the loop assuming that the architecture has forwarding and stalling hardware but that the compiler **does not** reorder the instructions:

ALU/branch pipeline	Load/store pipeline	
addi	lw	Cycle 1
	lw	Cycle 2
		Cycle 3
addi		…
add	sw	…
beq		…
		…
		…

b) It is possible to improve the efficiency of the code on the superscalar (without affecting the correctness of the code) by reordering the instructions. Specifically, by moving instruction L_5_ to the position between L_3_ and L_4__.

c) Show the pipeline schedule after you reorder the instructions as specified in part b.

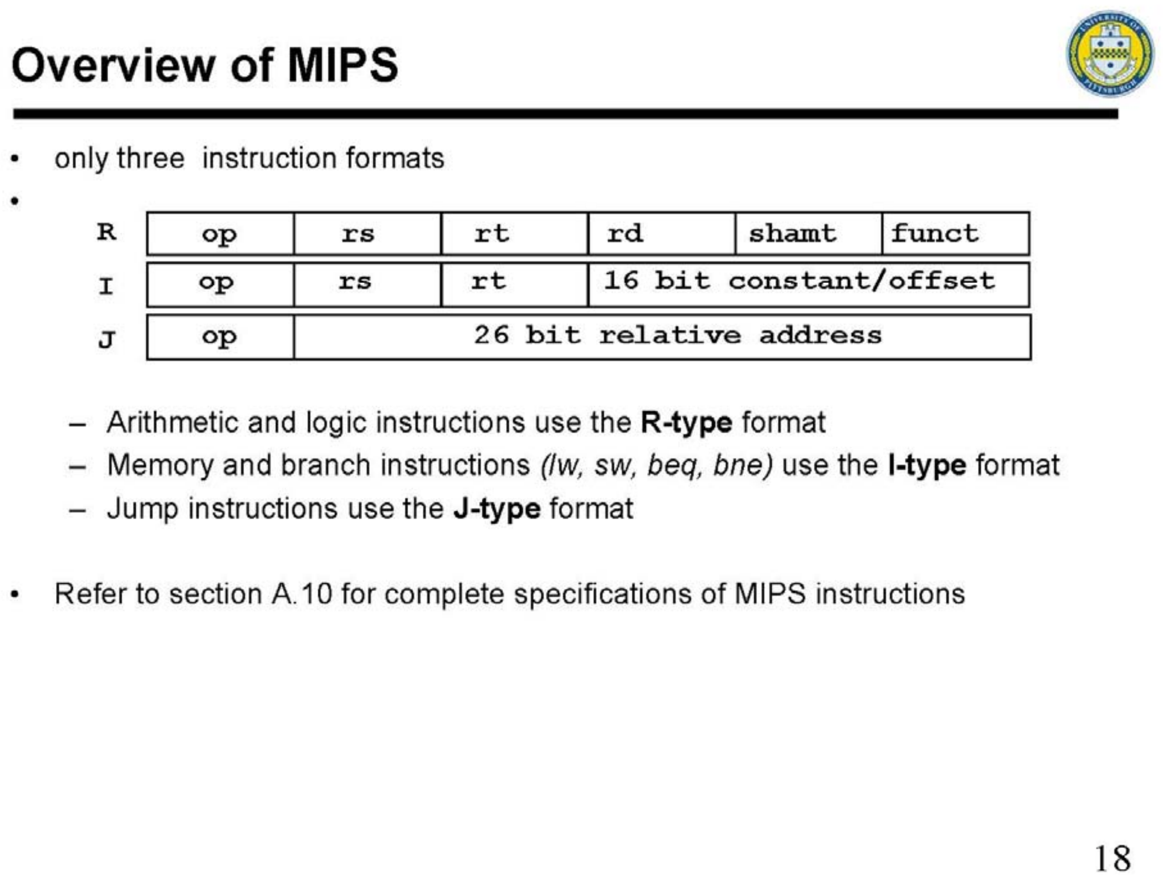
ALU/branch pipeline	Load/store pipeline	
addi	lw	Cycle 1
	lw	Cycle 2
add		Cycle 3
addi	sw	…
beq		…
		…
		…
		…

The following slide (from lecture 1) shows the machine instruction formats for the three types of instructions:

Rtype rd, rs, rt

I-type rt, rs (I)

J-type L



Overview of MIPS

- only three instruction formats
-

R	op	rs	rt	rd	shamt	funct
I	op	rs	rt	16 bit constant/offset		
J	op	26 bit relative address				

  - Arithmetic and logic instructions use the **R-type** format
  - Memory and branch instructions *(lw, sw, beq, bne)* use the **I-type** format
  - Jump instructions use the **J-type** format

- Refer to section A.10 for complete specifications of MIPS instructions

18