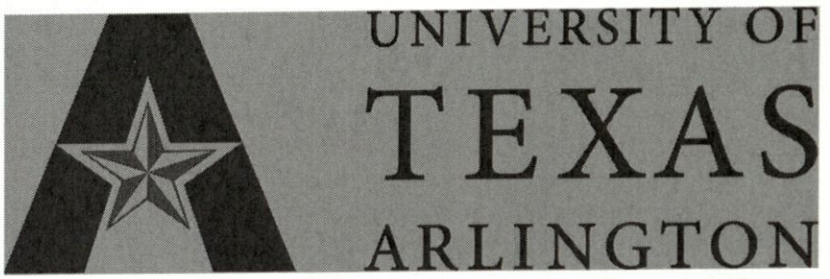


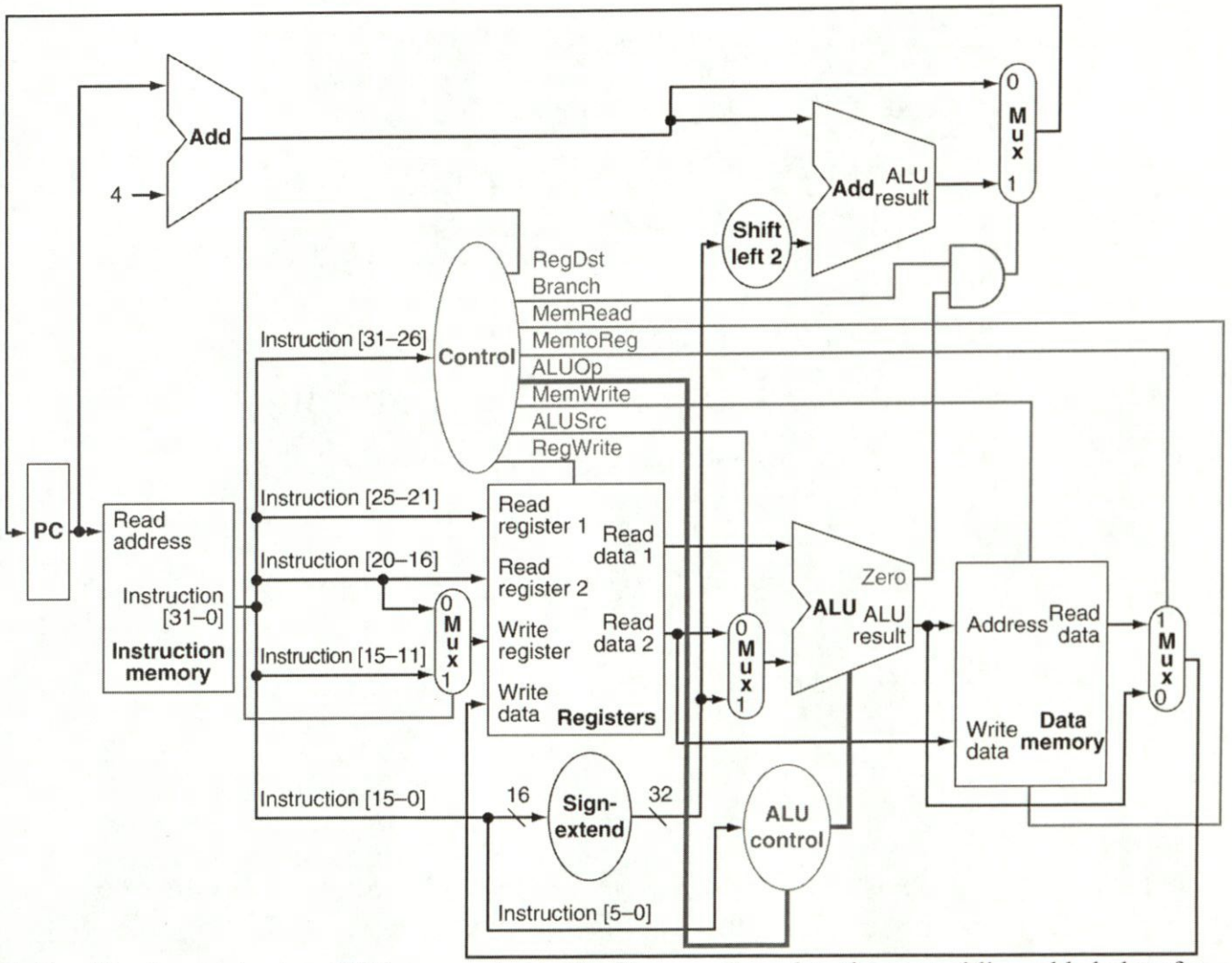
CSE 2312: Computer Organization & Assembly Language Programming Spring 2018

Homework #3

Student Name: Andrew Duong

Student ID: 1000867697

Directions: Answer the questions on the following pages. Show all applicable steps for any problems requiring the use of formulas or calculations. Submit your completed assignment electronically as a single PDF document with this completed coversheet as the first page and your name written at the top of all additional pages. You may also submit the document in person before the deadline, in which case this coversheet must be completed and stapled to your solution pages.



1. Consider the single stage CPU represented in the diagram. Complete the control line table below for the given instructions by entering 0, 1, or x for "don't care".

```
Instruction #1:     sltu $t0,$t1,$t2
Instruction #2:     10101100011000100000000000010100
Instruction #3:     sh $t0,$t1,100
Instruction #4:     beq  $t0,$t1,L1
Instruction #5:     0x0B090001
```

	RegDst	Branch	MemRead	MemtoReg	MemWrite	ALUSrc	RegWrite
#1	1	0	0	0	0	0	1
#2	X	0	0	X	0	0	0
#3	10	X	0	X	0	XX	1
#4	X	0	0	X	0	0	0
#5	X	0	1	X	X	0	1

Control unit is handling the control over the instruction over each instruction executed properly.

2. For each instruction in part 1, what would be the the values of the *ALUop* and *ALU control input* lines? Be sure to include only the proper number of bits for each.

3. Suppose the logic blocks in a processor have the following latencies...

Instruction fetch	Register read	ALU operation	Data access	Register write
350ps	150ps	200ps	450ps	200ps

a) In a single cycle, non-pipelined processor, what is the minimum time between instructions for an application executing only R-type instructions?

b) In a single cycle, non-pipelined processor, what is the minimum time between instructions for an application executing R, I, and J-type instructions?

c) If the logic blocks above are each implemented as individual pipeline stages, what would be the minimum time between instructions for this pipelined CPU if hazards are ignored?

4. Consider the following sequence of instructions executed on the basic 5 stage pipeline...

```
add  $t3, $t2, $t1
or   $t5, $t3, $t4
and  $t6, $t7, $t4
add  $t7, $t2, $t1
```

a) Assuming our processor has no forwarding or hazard detection, insert a minimal amount of pipeline stalls to ensure correct execution (you may stall the pipeline with the instruction `add $zero, $zero, $zero`).

b) Assuming our processor has no forwarding or hazard detection, rearrange the instructions and add stalls only if necessary to ensure proper execution. The values contained in the registers after executing your modified code should be equivalent to the unmodified execution.

5. Consider the following simple program executed on a pipelined MIPS CPU

```
        addi $s0, $zero, 3
LOOP:   lw $t0, 0($s1)
        add $t1, $t0, $t2
        add $t3, $t1, $tt4
        addi $s0, $s0, -1
        bne $s0, $zero, LOOP
        addi $s1, $zero, $zero
```

a) How many cycles will be required for the program to fully execute on a 5 stage pipeline with forwarding and perfect branch prediction?

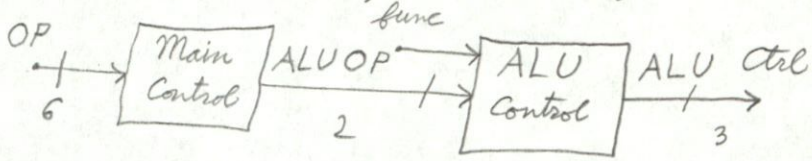
b) How many cycles will be required for the program to fully execute on a 5 stage pipeline that has no forwarding or branch prediction, but automatically inserts a minimal number of stalls?

(2.)

ALU Control input	Function	Operations
000	AND	and
001	OR	OR
010	Add	add, lw, sw
110	Subtract	sub, beq
111	SH	SH

→ ALU Opcodes.

00 – lw, sw, 01 – beq, 10 – R-format



→ Generating ALU Control

Instruction Opcode	ALUop	Instruction operation	Function Code	Defined ALU Action	ALU Control Input
lw	00	Load word	XXXXXX	add	010
sw	00	store word	XXXXXX	add	010
beq	01	branch eq	XXXXXX	subtract	110
R-type	10	add	100000	add	010
R-type	10	subtract	100010	subtract	110
R-type	10	AND	100100	and	000
R-type	10	OR	100101	or	001
R-type	10	SH	101010	SH	111

(3.) a) R-type instructions are completed in 4th step (i.e. Data access) so 5th step. Register write step is not required for R-type instructions. So, application executing R-type instruction need 4 steps only.

So, minimum time between instructions = instruction fetch history + register read attency + ALU operation latency + data access latency

$= 350ps + 175ps + 500ps = \boxed{1175ps}$

b) Application executing R, I and J type instruction needs all steps.

So, minimum time between instructions = instruction fetch latency + Register read attency + ALU operation latency + data access latency + Register write latency.

$= 350ps + 150ps + 175ps + 500ps + 200ps = \boxed{1375ps}$

(3.)

c) Pipelining reduces the cycle time to longest stage latency.
Hence the longest stage/step is data access which has latency.
So, minimum time between instructions for this pipeline CPU if hazards are ignored is latency of data access i.e. **500ps**

④ a)

```
add $t3, $t2, $t1
nop
nop
or  $t5, $t3, $t4
andi $t6, $t7, $t4
nop
nop
add $t7, $t2, $t1
```

b)

```
add $t3, $t2, $t1
andi $t6, $t7, $t4
nop
or  $t5, $t3, $t4
nop
add $t7, $t2, $t1
```

(5.) a)

5-stage pipeline with forwarding and branch prediction:
Execution:

Instruction	1	2	3	4	5	6	7
add $s0, $zero, 3	F	D	X	M	W		
LOOP: lw $t0, 0($s1)		F	D	X	M	W	
add $t1, $t0, $t2			F	D	X	M	W
add $t3, $t1, $tt4				F	D	X	M
addi $s0, $s0, -1					F	D	X
bne $s0, $zero, LOOP						F	D
addi $s1, $zero, $zero							F

The number of cycles for the MIPS program on 5 stage pipeline with forwarding and branch prediction is 7.
Instruction is fetched on every cycle of the 5 stage pipeline.

(5.) b)

5 stage pipeline without forwarding

Execution:

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13
addi $s0, $zero, 3	F	D	X	M	W								
LOOP: lw $t0, 0($s1)		F	D	X	M	W							
add $t1, $t0, $t2			F	D	X	M	W						
add $t3, $t1, $tt4				F	D	X	M	W					
addi $s0, $s0, -1					F	D	X	M	W				
bne $s0, $zero LOOP						F	D	-	-	X	M	W	
addi $s1, $zero								F	D	X	D	M	W
$zero													

Here, number of cycles extended without forwarding procedure in execution.

So, the number of cycles required is 13.