REFERENCE PROGRAM

All scheduling problems will refer to various executions and schedules of the following code:

```
Label:
lw      $4, 0($2)       # op 1
lw      $5, 0($3)       # op 2
add     $4, $4, $5      # op 3
sw      $4, 0($2)       # op 4
lw      $2, 4($2)       # op 5
lw      $3, 4($3)       # op 6
sltiu   $4, $2, 1       # op 7
sltiu   $5, $3, 1       # op 8
nor     $4, $4, $5      # op 9
bne     $0, $4, Label   # op 10
add     $4, $2, $3      # op 11
sw      $4, 4($2)       # op 12
```

REFERENCE MACHINE

All problems will assume the following ***unless otherwise stated:***

- Pipeline structure: FE-DE-RE-DI [– IS-WB –] CO
- Assume no older instructions exist in the pipeline (cycle 1 is Fetch from Label)
- Assume all branches correctly predicted
- Assume that the data structures pointed to by $2 and $3 are non-overlapping
- Instruction Cache: Assume all instructions are in the L1 I$ with hit time 1 cycle
- Data Cache: Assume all referenced data is in D$ with hit time 1 cycle
- LSQ scheduling: **Aggressive Speculation**
- Store buffer: Assume the store buffer does not run out of room and writes to memory in the first cycle that a memory port is idle during or after commit.
- Width: Fetch, Decode, Rename, Dispatch, Issue, WB, and Commit are all ***4-wide***. Assume a bypass-enabled queue exists between Decode and Rename and between Rename and Dispatch. Assume that neither queue ever becomes full **(i.e. fetch and decode never stall).**
- Functional units/Structural limitations: Three data memory ports, four WB busses, four Boolean/comparator ALUs, four integer ALUs (bne uses Boolean ALU).
- Assume no interrupts or exceptions occur
- Sizing: LSQ: 16 entry, IQ: 32 entry, ROB: 48 entry, Physical register file: 44 registers
- Assume that the initial mapping from architectural registers to physical registers is that AR i is in PR i for all architectural registers i, and that the free list is sorted and FIFO, i.e. **initial free list contents are [P32, P33, …P43]** and **initial map table contents are MapTable[$0]=P0, MapTable[$1]=P1,…MapTable[$31]=P31**
- **Operations that do not perform a writeback are finished after issue and are eligible for commit the next cycle**

Notes:

1. Register names cannot be consumed from the free list in the same cycle they are returned to it (they are available in the following cycle).
2. For conservative Load/Store scheduling, assume that the effective address computation occurs during issue – any instruction with a “maybe dependency” cannot issue until the following cycle.

Computer Engineering 431, Spring 2018 Exam 3, Tuesday, April 3rd

Exam time: 5 pre + 70 minutes exam

Test Value: 32 **pts. Total possible points:** 44 **(max score =** 137.5**%)**

Total = /32	P1: /32	P2: /4	P3: /4	P4: /4

Front Left Neighbor # _____	**Front Neighbor #** _____	**Front Right Neighbor #** _____
Left Neighbor # _____	**Your Name (print clearly):** _____	**Right Neighbor #** _____
Rear Left Neighbor # _____	**Rear Neighbor #** _____	**Rear Right Neighbor #** _____

REFERENCE SCHEDULE ALPHA

Original Instruction	Renamed INSTs				FE	DE	RE	DI	Issue	WB	Commit
Label: lw $4, 0($2)	**Label:** lw	P32 ← P2, 0		[P4]	1	2	3	4	5	6	7
lw $5, 0($3)	lw	P33 ← P3, 0		[P5]	1	2	3	4	5	6	7
add $4, $4, $5	add	P34 ← P32, P33		[P32]	1	2	3	4	6	7	8
sw $4, 0($2)	sw	← P34, P2, 0		[X]	1	2	3	4	7	-	8
lw $2, 4($2)	lw	P35 ← P2, 4		[P2]	2	3	4	5	6	7	8
lw $3, 4($3)	lw	P36 ← P3, 4		[P3]	2	3	4	5	6	7	8
sltiu $4, $2, 1	sltiu	P37 ← P35, 1		[P34]	2	3	4	5	7	8	9
sltiu $5, $3, 1	sltiu	P38 ← P36, 1		[P33]	2	3	4	5	7	8	9
nor $4, $4, $5	nor	P39 ← P37, P38		[P37]	3	4	5	6	8	9	10
bne $0, $4, **Label**	bne	← P0, P39, **Label**		[X]	3	4	5	6	9	-	10
Label: lw $4, 0($2)	**Label:** lw	P40 ← P35, 0		[P39]	3	4	5	6	7	8	10
lw $5, 0($3)	lw	P41 ← P36, 0		[P38]	3	4	5	6	8	9	10
add $4, $4, $5	add	P42 ← P40, P41		[P40]	4	5	6	7	9	10	11
sw $4, 0($2)	sw	← P42, P35, 0		[X]	4	5	6	7	10	-	11
lw $2, 4($2)	lw	P43 ← P35, 4		[P35]	4	5	6	7	8	9	11
lw $3, 4($3)	lw	___ ← ___, 4		[___]	4	5					
sltiu $4, $2, 1	sltiu	___ ← ___, 1		[___]	5	6					
sltiu $5, $3, 1	sltiu	___ ← ___, 1		[___]	5	6					
nor $4, $4, $5	nor	___ ← ___, ___		[___]	5	6					
bne $0, $4, **Label**	bne	← ___, ___, **Label**		[X]	5	6				-	
add $4, $2, $3					6	7					
sw $4, 4($2)					6	7					

Using **REFERENCE SCHEDULE ALPHA**, answer the following questions:

1. (32 pts) *Dynamic Scheduling*
   a) (8pts) Fill in all missing entries in reference schedule ALPHA.

   b) (4pts) If the next instruction after op 12, fetched in cycle 6, was **xor $7, $5, $2**, what would the renamed instruction be, and in what cycles would it dispatch, issue, writeback, and commit?

Renamed INST	Dispatch	Issue	WB	Commit

   c) (4pts) What are the contents of the free list after the **end** of cycle nine?

Free List (Head of queue (oldest) on left, tail at right, i.e. older→newer)

   d) (4 pts) What are the register mappings for $4 and $3 at the **end** of cycle six?
   $4 mapped to: $3 mapped to:

   e) (8 pts) Schedule the instructions below assuming **conservative** LSQ ordering.

SCHEDULE IOTA

Original Instruction	Renamed INSTs			FE	DE	RE	DI	Issue	WB	Commit
Label: lw $4, 0($2)	**Label:** lw	P32 ← P2, 0	[P4]	1						
lw $5, 0($3)	lw	P33 ← P3, 0	[P5]	1						
add $4, $4, $5	add	P34 ← P32, P33	[P32]	1						
sw $4, 0($2)	sw	← P34, P2, 0	[X]	1						
lw $2, 4($2)	lw	P35 ← P2, 4	[P2]							
lw $3, 4($3)	lw	P36 ← P3, 4	[P3]							
sltiu $4, $2, 1	sltiu	P37 ← P35, 1	[P34]							
sltiu $5, $3, 1	sltiu	P38 ← P36, 1	[P33]							
nor $4, $4, $5	nor	P39 ← P37, P38	[P37]							
bne $0, $4, **Label**	bne	← P0, P39, **Label**	[X]							

f) (4 pts) If there were only 36 physical registers, in what cycle would the (first) execution of op 7 **issue**? Why?

2. (4 pts) *In-order superscalar scheduling* [Revisited]
Schedule 1 iteration of the code from page 1 (repeated below) assuming a 4-wide, in-order superscalar processor with pipeline F-D-E-M-W, 3 data-memory ports, and 4 of all other resources. Assume that no instruction can advance to a pipeline stage in which any other instruction is currently stalled, that all branches are correctly predicted, all cache accesses are hits, and that no exceptions occur.

CYCLE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Label: lw $4, 0($2)	F																		
lw $5, 0($3)	F																		
add $4, $4, $5	F																		
sw $4, 0($2)	F																		
lw $2, 4($2)																			
lw $3, 4($3)																			
sltiu $4, $2, 1																			
sltiu $5, $3, 1																			
nor $4, $4, $5																			
bne $0, $4, **Label**																			

3. (4 pts) *Caching in Virtual Memory Systems [revisited, again]*

a) (1) The register transfer logic for "ADDU $rd, $rs, $rt" is written REG[$rd]=REG[$rs]+REG[$rt] considering the register file as the array REG. If we additionally consider memory as the byte-array MEM, what is the register transfer logic for "SW $rt, imm ($rs)"?

b) (3) Assume that you have a system with the following properties and configuration

(Same configuration as exams 1&2):

- The system is byte-addressable with 16-bit words
- Physical address space: 10 bits; Virtual address space: 16 bits; Page size: 2^4 bytes
- VIPT L1 D-cache = 48 bytes, 3-way associative, with 8-byte blocks; write-allocate/write-back
- Fully associative 3 entry DTLB; both DTLB and L1 D-Cache use LRU policy.
- Entry for each TLB or Cache entry consists of {valid, dirty, LRU-rank (00=most recent), tag, data}
  All metadata is given in binary. TLB data is in binary and Cache data is in hexadecimal.
- Endianness: If the data block containing address 0x0004 was 0x0123456789ABCDEF, the word loaded from 0x0004 would have integer value = 0x89AB.

DTLB:

1,0,00, 0000 1000 0000, 00 1100	1,0,10, 1101 0000 1101, 00 0001	1,0,01, 0001 1101 0000, 01 1111

L1 D-Cache:

SET 0	1, 0, 01, 00 1100, 0x1234567887654321	1, 0, 00, 00 0001, 0xCD9CEF0990FEDCBA	0, 0, 10, 01 1001, 0xBAD1BAD2BAD3BAD4
SET 1	1, 0, 10, 01 1111, 0xFEEDD0D0EEC5F00D	1, 0, 00, 00 1100, 0x1FEEDEE15DEADC0D	1, 0, 01, 00 0001, 0x0102030405060708

Given the initial contents of the DTLB and L1 D$ as shown, fill in the register value after the instruction executes.

LW $1, 0xD0DC($0); REGISTER_FILE[$1]=0x_____

4. (4 pts) *Potpourri*

a) (2 pts) Treating the structures pointed to by $2 and $3 as its arguments, i) what type of data structure is pointed to by each pointer, and ii) what does the reference program on page 1 do with these two structures?

i)

ii)

b) (2 pts) Assuming the availability of functions pNum() that prints the value in $a0, pFizz() and pBuzz() that print "fizz" and "buzz," respectively, and pN() that prints a newline, as well as pseudoinstruction MOD $rd, $rs, $rt that sets Reg[rd]=Reg[rs]%Reg[rt], write a MIPS assembly loop that for all integers N in [0,99] will print fizz for multiples of 3, buzz for multiples of 5, fizzbuzz for multiples of both 3 and 5, and N otherwise, each on a single line of output.

MAP TABLE AND FREE LIST WORKSHEET

ARCH.	PHYSICAL REGISTER MAPPING, By Cycle																
REG	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0																	
2																	
3																	
4																	
5																	

Free List

ARCH.	PHYSICAL REGISTER MAPPING, By Cycle																
REG	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0																	
2																	
3																	
4																	
5																	

Free List

ARCH.	PHYSICAL REGISTER MAPPING, By Cycle																
REG	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0																	
2																	
3																	
4																	
5																	

Free List
