

Computer Organization Exam 2 2020/5/25 (Closed book, Chapter 3-4.7)

(It's a closed book exam. All electronic devices must be turned off. Be sure to answer the questions in order. You will lose at least 5 points for each violation.)

1. (5%) What is $4363 + 3412$ when these values represent signed 12-bit octal numbers stored in sign-magnitude format? The result should be written in octal. Show your work.
2. (5%) Assume A and B are integers in two's-complement format. Explain how to detect overflow when executing $A+B$ (b) $A-B$ (hint: what are (1), (2), (3), and (4) in the following table).

Operation	A	B	Result indicating overflow
$A+B$	≥ 0	≥ 0	(1)
$A+B$	< 0	< 0	(2)
$A-B$	≥ 0	< 0	(3)
$A-B$	< 0	≥ 0	(4)

3. (10%) Answer the following question about IEEE-754
 - a. How to represent "Not-a-Number"? What situation should "Not-a-Number" be used?
 - b. What situation will a denormalized number in IEEE 754 be used in?
 - c. Does floating point number addition have associativity property? If not, explain why?
4. (15%) IEEE 754-2008 contains a half precision that is only 16 bits wide. The leftmost bit is still the sign bit, the exponent is 5 bits wide and has a bias of 15, and the fraction is 10 bits long. A hidden 1 is assumed. Answer the following questions:
 - a. (5%) Write down the bit patterns to represent 2.6375×10^1 assuming the half precision format is used to store the exponent.
 - b. (10%) Calculate the sum of 2.6375×10^1 and $4.150390625 \times 10^{-1} (=1.10101001 \times 2^{-2})$ by hand, assuming A and B are stored in the 16-bit half precision. Assume 1 guard, 1 round bit, and 1 sticky bit, and round to the nearest even. Show all the steps. Show the result in binary format
5. (10%) Design an optimize multiplication hardware based on the unoptimized hardware shown in Figure 1. Explain the difference between the unoptimized and optimized multiplier. The more optimizations and less hardware cost it has, the higher score you will get.

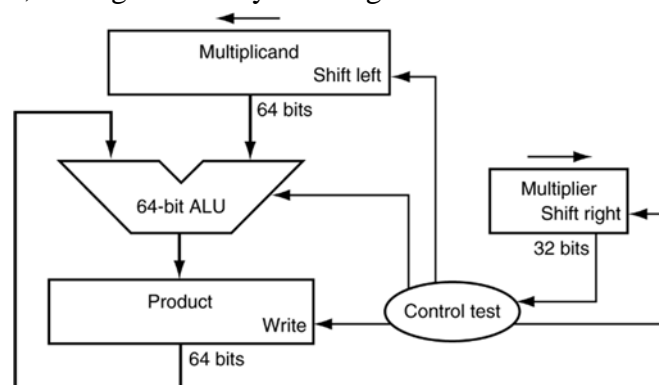


Figure 1. Unoptimized multiplication hardware

6. (10%) This problem examines how latencies of individual components of the datapath affect the clock cycle time of the entire datapath, and how these components are utilized by instructions. Assume the

following latencies for logic blocks in the datapath:

I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-Extend	Shift-Left-2
750ps	200ps	50ps	250ps	300ps	500ps	100ps	5ps

- What is the clock cycle time if the only types of instructions we need to support are ALU instructions (ADD, AND, etc.)?
- What is the clock cycle time if we only have to support LW instructions?

7. (5%) In a 5-stage pipelined processor, assume that there are no pipeline stalls and that the breakdown of executed instructions is as follows, what fraction of all cycles is the data memory used?

add	addi	not	beq	Lw	sw
20%	20%	0%	30%	20%	10%

8. (5%) Explain how many stalls are there in the following situation if forwarding is used? Explain why
- lw \$2, 20(\$1); sw \$2, 4(\$3)

9. (20%) In a 5-stage pipelined processor, assume each stage of the datapath has the following latency:

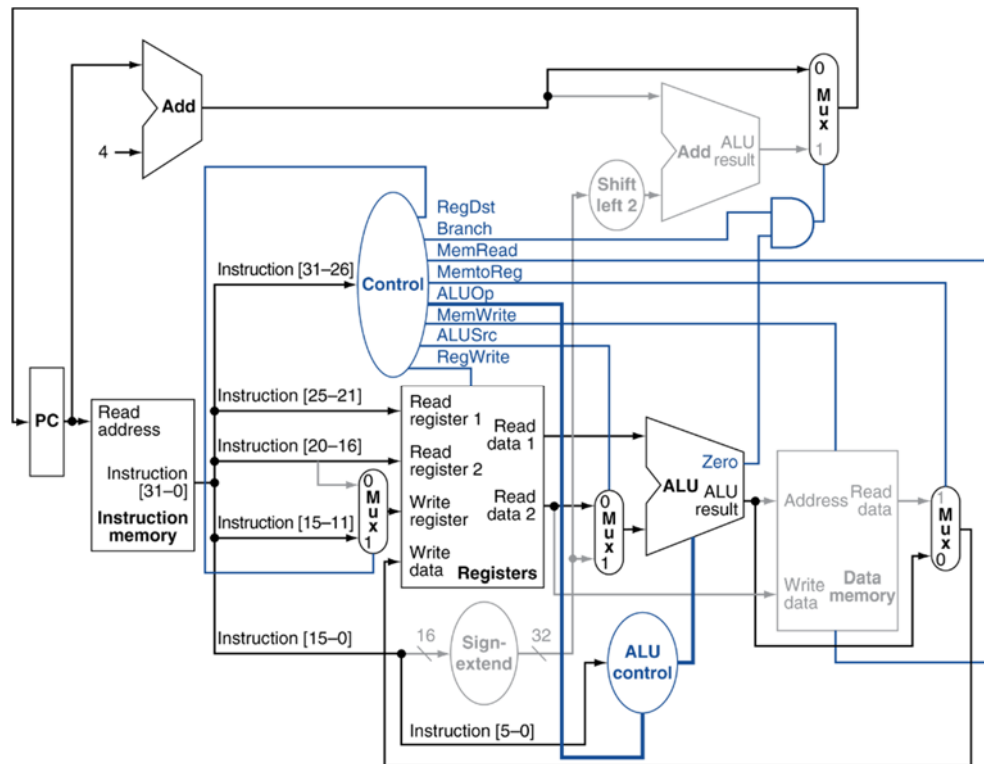
IF	ID	EX	MEM	WB
250ps	350ps	150ps	400ps	200ps

Also, assume that instructions executed by the processor are broken down as follows:

Alu	beq	lw	Sw
45%	20%	20%	15%

- What is the clock cycle time in a pipelined and non-pipelined processor?
 - What is the total latency of an LW instruction in a pipelined and non-pipelined processor?
 - If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?
 - Assuming there are no stalls or hazards, what is the utilization of the data memory?
 - Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Registers" unit?
10. (15%) Assume the CPU has the basic five-stage pipeline. The current instructions are
- lw \$1,40(\$2) ;
add \$2,\$3,\$3;
add \$1,\$1,\$2 ;
sw \$1,20(\$2)
- List the instructions that have dependences, and indicate the type of dependences.
 - Assume there is no forwarding in this pipelined processor. Indicate hazards and add nop instructions to eliminate them.
 - Assume there is full forwarding. Indicate hazards and add nop instructions to eliminate them.

11. (30%) The following figure is a single-cycle implementation. Different instructions utilize different hardware blocks in this implementation.



a. (10%) Draw the datapath of the *addi* instruction. Remove blocks and wires that are not used in the *addi* instruction

Ans: Addi is I-type instruction

b. (14%) Complete the mapping table in Figure 2 for the control unit. What are the values of (1)-(14)?

	Signal	R-type	lw	sw	beq
Instruction	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Control	RegDst	(1)	(2)	X	X
	ALUSrc	(3)	(4)	1	0
	MemtoReg	(5)	(6)	X	X
	RegWrite	(7)	(8)	0	0
	MemRead	(9)	(10)	0	0
	MemWrite	(11)	(12)	1	0

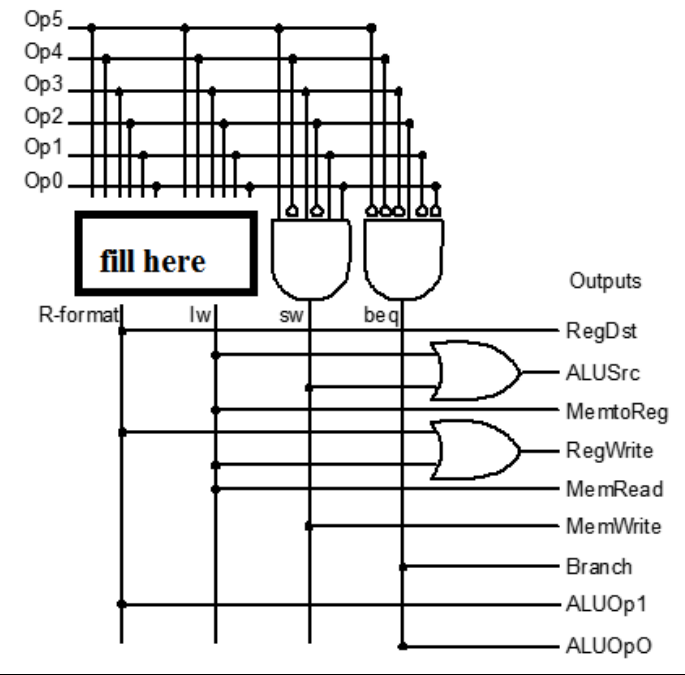
Branch	(13)	(14)	0	1	Inputs Op5 Op4 Op3 Op2 Op1 Op0 
ALUOp1	1	0	0	0	
ALUOp2	0	0	0	1	

Figure 2. Input-output table for Control

Figure 3: circuit for control

c. (6%) Complete the circuit design in Figure 3 for the control unit.

a. See the data path

• Use PLA

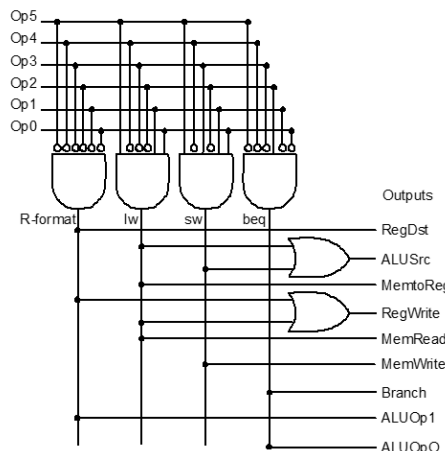
Signal name	R-format	lw	sw	beq
Op5	0	1	1	0
Op4	0	0	0	0
Op3	0	0	1	0
Op2	0	0	0	1
Op1	0	1	1	0
Op0	0	1	1	0
RegDst	1	0	x	x
ALUSrc	0	1	1	0
MemtoReg	0	1	x	x
RegWrite	1	1	0	0
MemRead	0	1	0	0
MemWrite	0	0	1	0
Branch	0	0	0	1
ALUOp1	1	0	0	0
ALUOp0	0	0	0	1

Truth table for main control signals

Main control PLA (programmable logic array)

$$RegDst = \overline{Op5} \cdot \overline{Op4} \cdot \overline{Op3} \cdot \overline{Op2} \cdot \overline{Op1} \cdot \overline{Op0}$$

ALUSrc=?



b. &c.

12. (10%) Assume A and B are integers in two's-complement format. Explain how to detect overflow when executing A+B (b) A-B (hint: complete the following table).

Operation	A	B	Result indicating overflow
A+B	≥ 0	≥ 0	?
A+B	< 0	< 0	?
A-B	≥ 0	< 0	?
A-B	< 0	≥ 0	?

You may refer to the following MIPS instruction encoding table.

Op(31:26)								
28-26 31-29	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	R-format	Bltz/gez	Jump	Jump & link	Branch eq	Branch ne	blez	bgtz
1(001)	Add immediate	Addiu	Set less than imm.	Set less than imm. Unsigned	andi	ori	xori	Load upper imm.
2(010)	TLB	FlPt						
3(011)								
4(100)	Load byte	Load half	Lwl	Load word	Load byte unsigned	Load half unsigned	lwr	
5(101)	Store byte	Store half	Swl	Store word			swr	
6(110)	Load linked word	lwcl						
7(111)	Store cond. word	swcl						

Op(31:26)=000000 (R-format), funct(5:0)								
2-0 5-3	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	Shift left logical		Shift right logical	sra	sllv		srlv	srav
1(001)	jump register	jalr			Syscall	Break		
2(010)	mfhi	mthi	mflo	Mtlo				
3(011)	Mult	Multu	div	Divu				
4(100)	Add	Addu	Subtract	Subu	And	Or	Xor	Not or (nor)
5(101)			Set l.t.	Set l.t. unsigned				
6(110)								

7(111)								
--------	--	--	--	--	--	--	--	--

Name	Fields						Comments
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions are 32 bits long
R-format	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	op	rs	rt	address/immediate			Transfer, branch, imm. format
J-format	op	target address					Jump instruction format

Name	Register Number	Usage	Preserve on call?
\$zero	0	constant 0 (hardware)	n.a.
\$at	1	reserved for assembler	n.a.
\$v0 - \$v1	2-3	returned values	No
\$a0 - \$a3	4-7	Arguments	Yes
\$t0 - \$t7	8-15	temporaries	No
\$s0 - \$s7	16-23	saved values	Yes
\$t8 - \$t9	24-25	temporaries	No
\$gp	28	global pointer	Yes
\$sp	29	stack pointer	Yes
\$fp	30	frame pointer	Yes
\$ra	31	return addr (hardware)	Yes

13. (10%) The problem examines how latencies of individual components of the datapath affect the clock cycle time of the entire datapath, and how these components are utilized by instructions. Assume the following latencies for logic blocks in the datapath:

I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-Extend	Shift-Left-2
750ps	200ps	50ps	250ps	300ps	500ps	100ps	5ps

- c. What is the clock cycle time if the only types of instructions we need to support are ALU instructions (ADD, AND, etc.)?
- d. What is the clock cycle time if we only have to support LW instructions?

14. (10%) Explain how many stalls are there in the following two situations if forwarding is used? Explain why
- b. lw \$2, 20(\$1); and \$4, \$2, \$5
 - c. lw \$2, 20(\$1); sw \$2, 4(\$3)

(10%) This exercise examines the accuracy of various branch predictors for the following repeating pattern of branch outcomes: T, T, T, NT, NT

- a. What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?
- b. What is the accuracy of the two-bit predictor if this pattern is repeated forever?

-backup question ---