2025 Spring CS110 Final Exam Answer Sheets

Please write down ALL your answers on the answer sheets to get marked.

2. True or False 2points each

(a)	(b)	(c)	(d)	(e)	(f)	(g)
T	T	F	T	F	T	F
(h)	(i)	(j)	(k)	(l)	(m)	
T	T	T	F	F	T	

3. Cluster Computing

(a) Answer and explanation: 219000 hours. (1 point)

2025 has 365 days. Assume that MTTF for these SSDs is x. (1 point)

$$\frac{24 \times 365 \times 100}{100\,x} \leq 4\%. \Rightarrow x \geq 219000 \text{ hours}$$

(b)

i. Answer and explanation:

(1 point)

Yes, she can get 0x233. The program has a valid/correct/good/... lock and no data race occurs, (2 points)

ii. Answer and explanation:

(1 point)

Yes, for true sharing. For example, it happens to lock. (1point)

(1point)

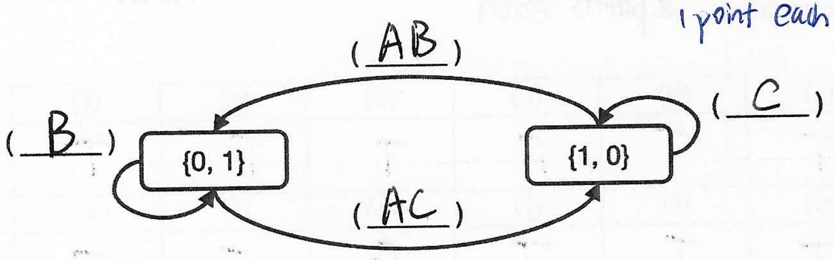
Yes. for false sharing as well. When one is modifying lock, false sharing may occur 'val' being handled by the other one. (1 point)

4. Digital circuit and finite-state machine (FSM)

(a) Complete the FSM state transistion diagram. Select all conditions that apply.

Fill in the blanks below the transition conditions

1 point each



(b)

Complete this truth table for the detector first and then build your circuit below

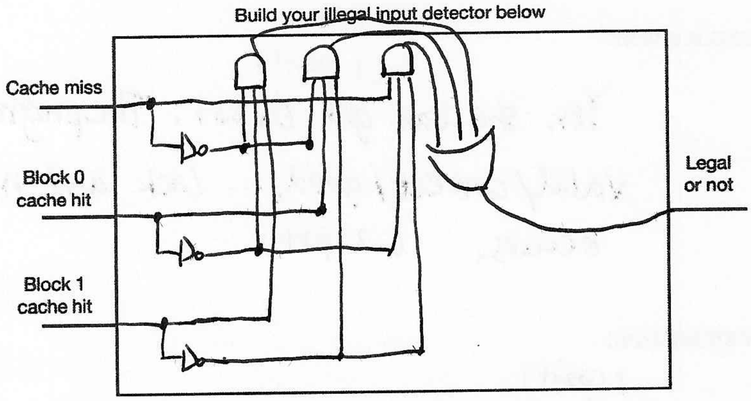
cache miss	block 0 cache hit	block 1 cache hit	Legal
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

3.5 points

0.5 per line

Build your illegal input detector below

1.5 points



The detector that you are required to build is a A (select all that apply)

1 point

A. digital circuit

B. synchronous circuit

C. register

(c) Complete the truth table for the LRU-2 manager below.

Complete this truth table for the **LRU-2 manager**

cache miss (M)	block 0 cache hit (hit0)	block 1 cache hit (hit1)	LRU$_n$ (L_n)	L_{n+1}
1	0	0	0	1
0	1	0	0	0
0	0	1	0	1
1	0	0	1	0
0	1	0	1	0
0	0	1	1	1

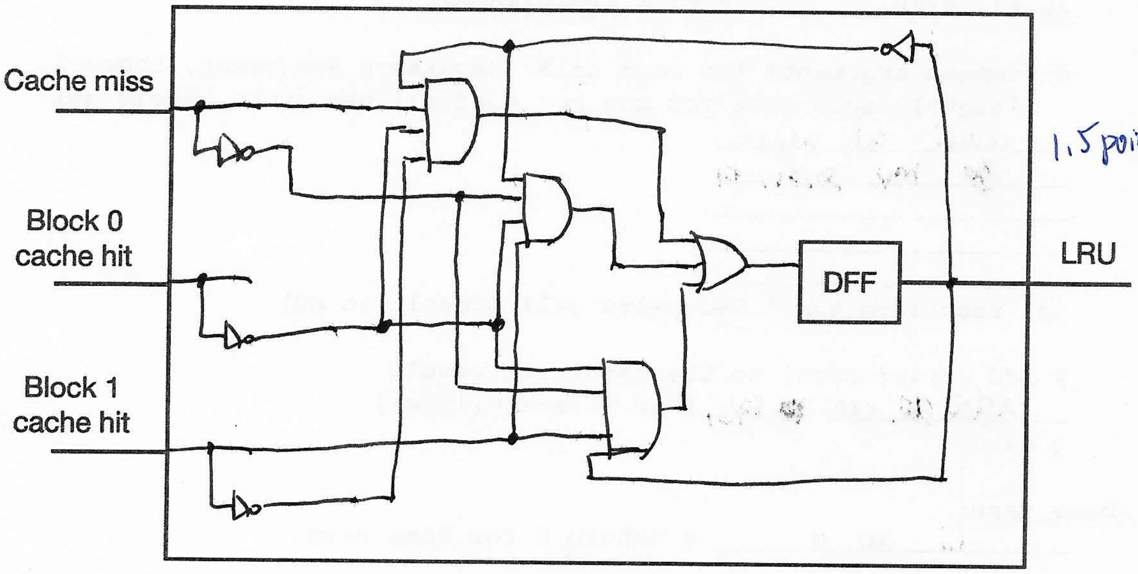
0.5 point each line

Write down the logic expression of the truth table in the form of **sum of minterm**. To simplify, use "M" to represent "cache miss", "hit0" for "block 0 cache hit", "hit1" for block 1 cache hit and "L" for LRU in the logic expression. Leave the logic expression as it is and do not simplify.

L_{n+1} = $M\,\overline{hit0}\,\overline{hit1}\,\overline{L_n} + \overline{M}\,\overline{hit0}\,\cancel{hit1}\,hit1\,\overline{L_n} + \overline{M}\cdot\overline{hit0}\cdot hit1\cdot L_n$ 2 points

Build your **LRU-2 manager** below with exactly the logic expression above. You can use ONLY `not`, `and` and `or` gates with arbitrary number of inputs.

Build your **LRU-2 manager** below



1.5 points

5. RISC-V

(a) Fill in the following RISC-V assembly code (assume on a 32-bit platform). Hints are provided at each instruction to be filled in:

```
recursive_sum:
    # Prologue: Save callee-saved registers and ra
    addi sp, sp, -12        # Allocate stack frame for needed regs
    sw ra, 8(sp)            # Save return address to stack
    sw s0, 4(sp)            # Save s0 (callee-saved) to stack
    sw s1, 0(sp)            # Save s1 (callee-saved) to stack

    # Base case: if index >= length, return 0 (use a branch instruction)
    bge a1, a2, base_case

    # Recursive case:
    addi s0, a0, 0          # s0 = array base address
    addi s1, a1, 0          # s1 = current index

    # Load array[index] into t0
    slli t0, a1, 2           # t0 = index * 4 (byte offset)
    add t0, s0, t0           # t0 = &array[index]
    lw t1, 0(t0)             # t1 = array[index]

    # Prepare arguments for next call: recursive_sum(array, index+1,
      length), note that you may not need all the lines before jal
    addi a1, a1, 1
    mv s0/s1, t1
    ________________
    ________________
    ________________
    jal recursive_sum # Recursive call (result in a0)

    # Add array[index] to the recursive result
    add a0, a0, s0/s1        # a0 += array[index]
    j exit

base_case:
    li a0, 0                 # Return 0 for base case

exit:
    # Epilogue: Restore saved registers and return
    lw s1, 0(sp)
    lw s0, 4(sp)
    lw ra, 8(sp)
    addi sp, sp, 12          # Deallocate stack
    ret
```

[redacted], 8 points in total

minus 0.5 for each wrong line until 0.

(b)

i. Answer and explanation:

For an array of length n. there are $n+1$ stack frames (one per recursive call + base case). Each frame uses 12 bytes (ra, s0, s1). (1point)

$(n+1) \times 12$ (1point)

ii. Answer and explanation:

The program would jump to garbage on ret or crashing or looping infinitly. One possible outcome /one explaination worths 1 point. Maximum 2 points.

6. Numbers, Pipeline, and Memory Hierarchy

(a) 2points

The Hamming ECC codeword is 0x5EB8 in the **hexadecimal** format (for this question you do not need to add one extra parity bit at the MSB).

(b)

i. From (A) to (F), put **the least** and **correct** `beqz` instruction(s) onto one or some of these places.

(A) ________________;

(B) ________________;

(C) beqz t2, 14; 1points

(D) ________________;

(E) ________________;

(F) ________________.

ii. List all hazards for I1 to I4 instructions in the table below.

Instructions	I1	I2	I3	I4
I1	-	C	C	C
I2	-	-	D	N
I3	-	-	-	N
I4	-	-	-	-

1 point each blank

(c) The cache hit rate is 40 % (in percentage, that is, $x\%$). 0.5 point

Please fill in the following tables.

Virtual address	PPN	Page hit/fault	Physical address
0x11C	0x02	hit	0x05C
0x120	0x01	fault	0x020
0x124	0x01	hit	0x024
0x0F8	0x03	fault	0x078
0x128	0x01	hit	0x028
0x12C	0x01	hit	0x02C
0x130	0x01	hit	0x030
0x120	0x01	hit	0x020
0x134	0x01	hit	0x034
0x020	0x05	falt	0x0A0

VPN	PPN (excluding valid bit)
0x00	
0x01	0x05
0x02	
0x03	
0x04	
0x05	
0x06	0x00
0x07	0x03
0x08	0x02
0x09	0x01
0x0A	0x04
0x0B	
0x0C	0x06
0x0D	
0x0E	0x08
0x0F	

1 point each
(in total 3 points)

minus 0.2 point
for modifying each lines
that should not be modified.
until 0.

Block Number	0	1	2	3
Tags (Initially)	0x0A	0x0B	0x0C	0x0D
Tags (Finally)	0x05	0x01	0x01	0x03

4 points

7. Datapath

(a)

i.

instrution	imm	reg_en	is_beq	op2	ALU	re	we	wb_src
addi	1	1	0	1	1	0	0	1
lw	1	1	0	1	1	1	0	0
add	X	1	0	0	1	0	0	1
beq	2	0	1	0	4	0	0	X
sw	0	0	0	1	1	0	1	X

1 point each line

ii.

instruction	A	B	C
addi	8	8	8
lw	-4	4	2
add	X	4	4
beq	-8	0	X
sw	4	8	X

1 point each line

(b)

i.

PC Reg	IMem	Ctrl.	Imm.	Regfile	MUX(op2)	ALU	DMem	MUX(wb)
20	820	1070	1190	1120	1260	1460	2260	2330

2 points in total.

ii. Answer:

PC → IMem → Control → Imm → Mux (op2) → ALU → DMem → Mux(wb) → regfile

2 points

iii. Answer:

$1 / (\text{clk-to-q} + t_{comb} + t_{setup})$.

2 points