

CS 350, Spring 2009: Midterm Exam **Soln** (60 min)

Instructions

The exam is open book, open notes (but no sharing books or notes) and no support equipment (calculators, phones, computers, etc). All the questions are short-answer. The usual penalty for copying or sharing answers on a quiz or exam is a final grade of E for the course. If you have any questions, please ask during the quiz, not after

For questions that ask for a brief explanation, a couple of sentences should be plenty. Questions 7–16 refer to the LC-3; for questions that say “write an instruction,” a 16-bit binary representation is enough (mnemonic descriptions and comments are optional).

Histogram of Scores

92 92 90 90 90

89 89 88 87 86 85 81 81 81

79 78 78 77 76 74 72

58 53

Avg 81.1

Stdev 10.1

Questions

- [8 points] What is the standard IEEE floating-point representation for 4.5_{10} ?

$4.5_{10} = 100.1_2 = 1.001_2 \times 2^2 = 1.001_2 \times 2^{129-127}$, so the IEEE representation is a one-bit sign of 0, an eight-bit exponent of 1000 0001, and 23-bit fraction of 001 (the leading 1 in 1.001 is dropped) followed by 20 0 bits: 0 1000 0001 001 0000 0000 0000 0000 0000.

- [9 points] Let $X = 10111$ and $Y = 11001$ be two 5-bit 2's complement integers. (a) Calculate $Z = X+Y$, (b) Say what X , Y , and Z are in decimal, and (c) Say whether or not overflow has occurred.

$X+Y=Z$ is $10111 + 11001 = 10000$ is $-9 + -7 = -16$; no overflow

- [8 points] Say we have an 8-bit value V . What logical operation(s) do we need to do to see if bit 5 is 0? (Assume bits are numbered 0–7 from right to left.)

Bit 5 = 0 iff $(V \text{ AND } 0010\ 0000)$ is zero.

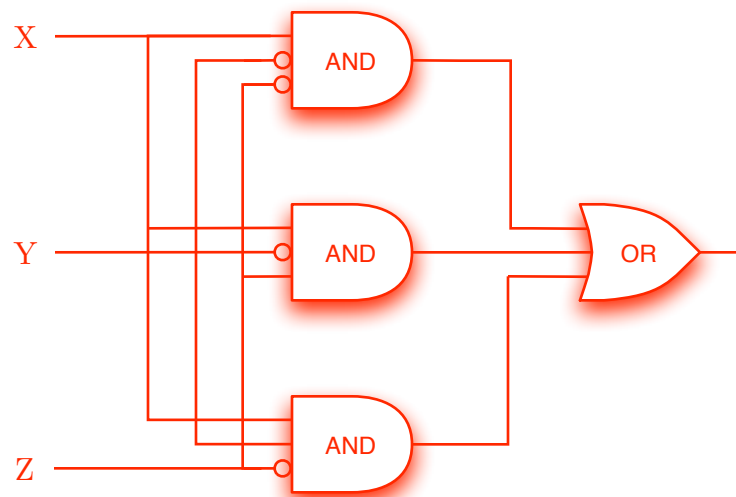
- [8 points] Do combinatorial logic circuits require clocks? Briefly, why or why not? They don't need clocks. Their outputs are purely a function of their inputs, so there's no internal state whose change you need to synchronize with the calculation; you just need to wait until the computation settles down. [The wait time comes from the length of the cir-

cuit. Someone pointed out you could use a clock if you wanted to keep track of how long you've waited.]

5. [8 points] Draw a truth table for $(X \text{ OR } Y) \text{ AND } (\text{NOT } Y \text{ OR } (X \text{ AND NOT } Z))$.
Next time, please list the XYZ values in the order below: from 000 to 111.

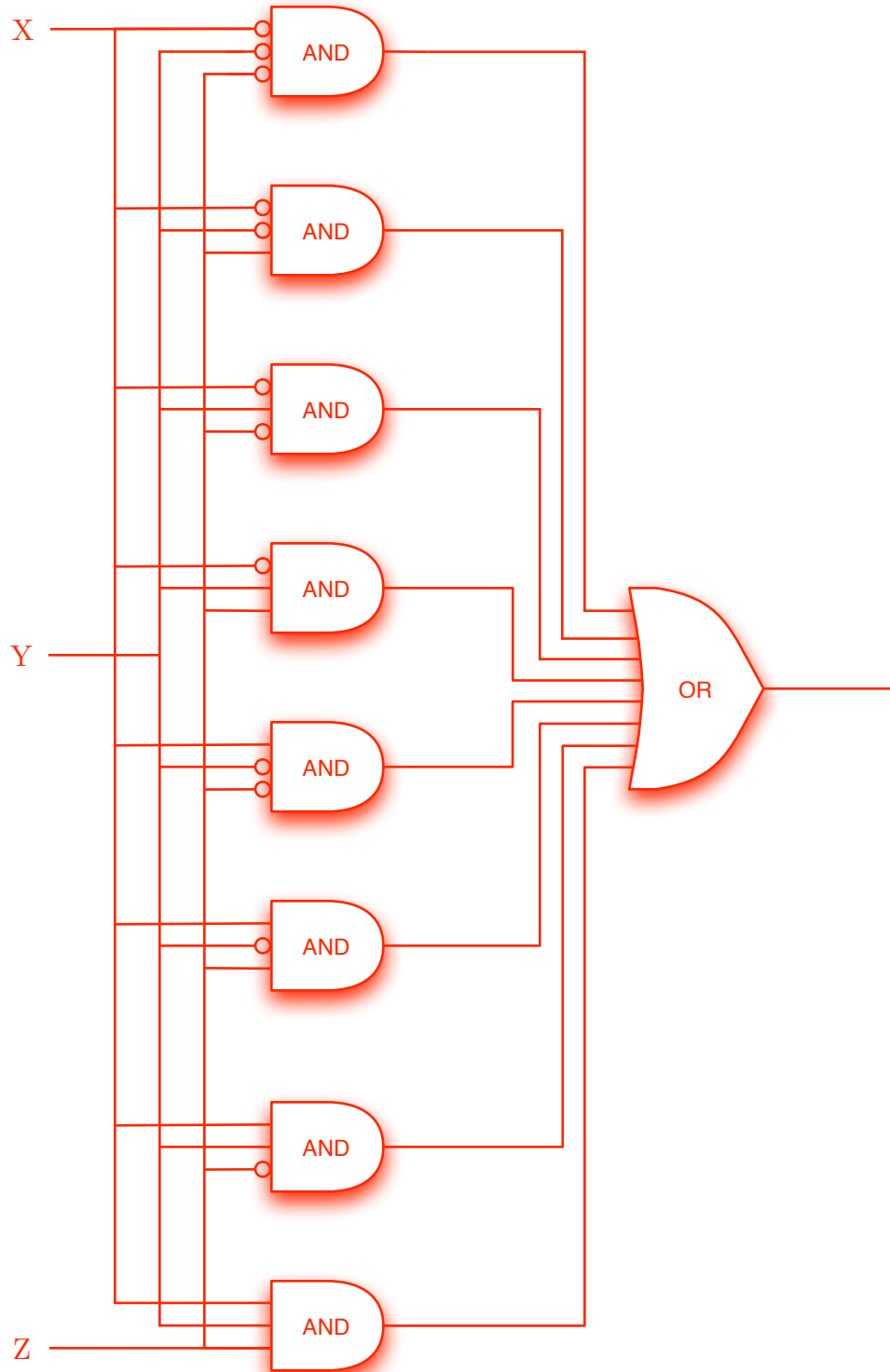
X	Y	Z	X OR Y	NOT Y OR (X AND NOT Z)	(X OR Y) AND (NOT Y OR (X AND NOT Z))
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	0	0

6. [9 points] Draw a logic gate implementation for your truth table from the previous question. You must use the technique that uses AND gates whose output goes to a single OR gate. (The AND gates must take 3 inputs: X or NOT X, Y or NOT Y, and Z or NOT Z.)



How to solve this kind of problem in general: Take the logic diagram below — it has 8 AND gates, one for each row of the XYZ table (the top gate is for

$X=Y=Z=0$, the next one is for $X=Y=0, Z=1$, ... the bottom gate is for $X=Y=Z=1$). Find the gates that correspond to the truth table rows with entries of 1; keep those gates and throw away the others.



7. [5 points] During what phase of the instruction cycle does a branch instruction cause the branch to occur, and how does it do it?

During the execute instruction phase, by setting the program counter to the target address: If $(N\bar{Z}P \text{ mask AND } N\bar{Z}P \text{ condition codes}) \neq 0$, then $PC \leftarrow PC + \text{sign-extended offset}$.

8. [5 points] Do you expect the LD and LDI instructions to take the same amount of time or different amounts of time? Explain briefly (a couple of sentences is plenty).

Different amounts of time: the LD instruction makes one reference to memory and the LDI instruction makes two references, and memory references take a long time.

9. [5 points] Write an LC-3 unconditional branch instruction that jumps to next+128 (or explain briefly why it's impossible).

BRN \bar{Z} P next+128 is 0000 111 0 1000 0000

10. [5 points] Write an instruction or instruction sequence that sets R5 to -3 (or explain briefly why it's impossible).

AND R5, R5, #0 is 0101 101 101 1 00000 (to set R5 to 0), followed by ADD R5, R5, #-3, which is 0001 101 101 1 11101

11. [5 points] Write an instruction that loads into R7 the value pointed to by R3 (or explain briefly why it's impossible). If it's helpful, assume the load instruction will be at location x3800.

LDR R7, R3, #0, which is 0110 111 011 000000

12. [5 points] Programs on a typical programmable calculator can access and modify the memory that contains data but not the memory that contains programs. Is such a calculator a von Neumann computer? Explain briefly.

No. A von Neumann computer treats instructions as data; the calculator above stores programs as data but cannot do calculations on instructions.

13. [5 points] If we want to copy the value at memory location L into R0, what actions do we have to take with the MAR and MDR?

MAR \leftarrow L; signal READ memory; R0 \leftarrow MDR

14. [5 points] Say registers R0 – R7 contain 12, -25, 8, -13, 34, -5, 16, 27, and the Z condition code is set. (You may not need all this information to solve the following question.) What happens if we execute 1001 1101 0111 1111 ?

Since $1001\ 110\ 101\ 111111$ is (NOT R6, R5) and $R5 = -5 = 1111\ 1111\ 1111\ 1011$, R6 becomes $0000\ 0000\ 0000\ 0100 = 4$. (Note: Since $(\text{NOT } -5)+1 = 5$, we must have that $(\text{NOT } -5) = 5-1 = 4$.)

15. [5 points] Write an instruction that loads into R0 the value stored at location L, where L is the contents of memory location x3900 (or explain briefly why it's impossible). If it's helpful, assume the load instruction will be at location x3850.

LDI R0, next+xAF, which is $1010\ 000\ 0\ 1010\ 1111$. We need xAF because then the effective address for the branch is $x3851+xAF = x3900$. (Remember, at the time we calculate the effective address, the PC has already been incremented from x3850 to x3851.) We need the LDI instruction because plain LD would set $R0 \leftarrow M[x3900] = L$; we want $R0 \leftarrow M[M[x3900]] = M[L]$. A surprising number of people made hex arithmetic errors. In particular, $x3850+x50 = x38A0$, not x3900.

16. [5 points] Which part of the LC-3 holds the current instruction being executed? The Instruction Register (IR). Some wrong answers: The PC (holds the address of the next instruction); the decoder (connects to the IR to figure out what instruction we have); the MAR (during Fetch, this holds the address of the instruction to retrieve); the MDR (during Fetch, we set the IR with this value).