Homework 15

Homework 15 is due on Sunday, May 24, by midnight. Remember to include your *Discussions section* (e.g. ED1) and follow the complete Homework submission guidelines.

Please ask all questions about this assignment during the office hours, or post them on piazza.

LC-3 control unit design; codes

1. Extending LC-3 ISA

You are adding a new instruction, called **CP**, to the LC-3 instruction set. This instruction copies a value from one memory location whose address is stored in the register specified by IR[11:9] to another memory location whose address is specified as a PC-relative offset by IR[8:0]. The opcode for this new instruction is 1101.

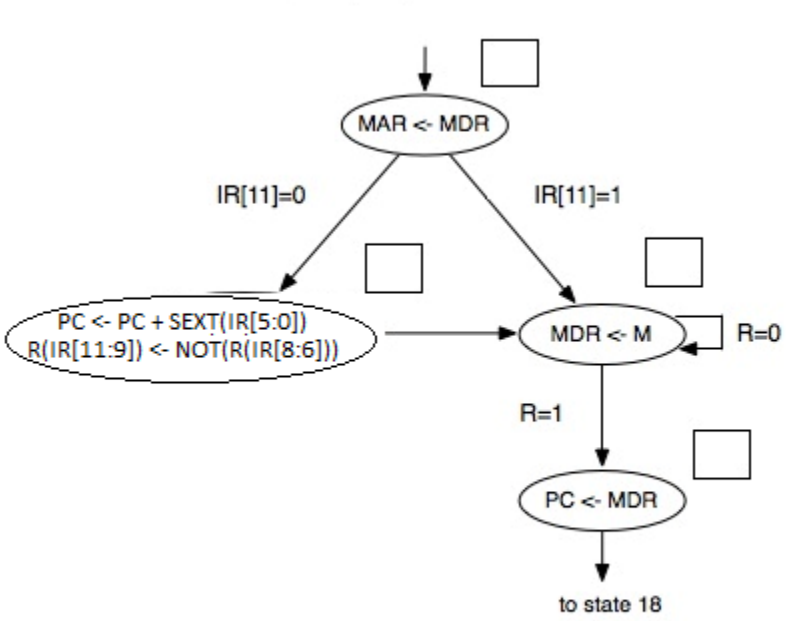
1. In RTL form, give a sequence of (at most 4) *microinstructions* that implement the execute phase of the **CP** instruction. Make sure your implementation does not modify any values in the general-purpose register file.
2. Determine control ROM microinstructions that implement the RTL statements from part (a). **Complete the table below by filling in 0, 1, or x as appropriate.** Use don't cares wherever possible. Specify ROM addresses in decimal. When you need *additional* states, state numbers **55**, **56**, **57**, and **58** are available for your use.

ROM address	IRD	COND(3)	J(6)	LD.BEN LD.MAR LD.MDR LD.IR LD.PC LD.REG LC.CC	GateMARMUX GateMDR GateALU GatePC	MARMUX PCMUX(2) ADDR1MUX ADDR2MUX(2) DRMUX(2) SR1MUX(2) ALUK(2)	MIO.EN R.W
				Do not fill in this space.			
				Only fill in control word bits for the first two microinstructions.			

3. We could have accomplished the task of copying a value from one memory location to another memory location by simply writing a short program using existing instructions. Write such a program in LC-3 assembly language assuming that the memory source address is stored in R6 and the memory destination address is labeled as DEST.

2. LC-3 FSM

The four states shown below are to be added to the LC-3 state diagram.



1. Assign state numbers and write them in the squares next to each state. Use only state numbers in the range 52-58.
2. Are there other possible correct answers (i.e., other state number assignments in the range 52-58)? Explain.
3. Determine the words in the control ROM corresponding to the two “middle” states in the above figure. Specify the ROM addresses used, and fill in the ROM content for each of these words. Enter 0, 1, or x values as appropriate: if the answer is ‘don’t care’, then you must use ‘x’.

ROM Adress	IRD	COND	J	LD(BEN,MAR,MDR,IR,PC, REG,CC)	GATE(MARMUX,MDR, ALU,PC)	MARMUX,PCMUX,ADDR1MUX,ADDR2MUX, DRMUXSR1MUX,ALUK	MIO.EN, R.W

3. LC-3 microsequencer

In this problem, you will design an alternative microprogrammed control unit for the LC-3. Each control address has 7 bits. The Execute Phase for the instruction whose opcode is in IR[15:12] begins at control address IR[15:12]||000 (i.e., IR[15:12] concatenated with 000). Each Execute Phase has at most 8 steps at consecutive control addresses. For example, for the ST instruction, whose opcode is 0011, the Execute Phase consists of 3 steps at control addresses 0011000, 0011001, and 0011010. After the last step, the control address of the Fetch Phase is loaded into the CAR.

The Fetch Phase begins at control address 1101000 (since opcode 1101 is not assigned). The following two steps of the Fetch Phase are at control addresses 1101001 and 1101010. The Instruction Decode step is at control address 1101011.

You will use a 7-bit control address register (CAR), implemented as an up-counter with parallel load. The CAR has an input LI (Load/Increment): when LI = 0, the CAR loads a new 7-bit address; when LI = 1, the CAR increments its binary value by 1. After a microinstruction is performed, the next microinstruction is at the next control address.

In place of the 10 bits of next address fields in the LC-3 microinstruction (IRD, COND, J), you will use a 3-bit field NEXT (NEXT[2:0]) with the meanings as given in the table below. The other 25 bits of the microinstruction define the control signals for the data path and remain the same as our ordinary LC-3.

NEXT	Next control address
000	IR[15:12]||000
001	CAR if R = 0, or CAR + 1 if R = 1
010	1101000 (go to Fetch) if BEN = 0, or CAR + 1 if BEN = 1
011	1101000 (go to Fetch)
1xx	CAR + 1

1. Write a minimal SOP Boolean expression for the LI input to the CAR. Use NEXT[2], NEXT[1], NEXT[0] for the individual bits of the NEXT field.
2. Show the control address and the value of the NEXT field for each microinstruction in the execute phase of the LDR instruction.
3. Trace through the fetch, decode, and execute phases of the BR instruction. Show the control address and the value of the NEXT field for all microinstructions.
4. Draw an implementation of the microsequencer, which determines the next control address. Your circuit will contain: the CAR; one multiplexer, demultiplexer, decoder, or encoder; AND, OR, and NOT gates - as few as possible.

4. Pair codes

A friend of yours is quite excited about having developed a new class of codes: pair codes. Pair codes are a generalization of the 2-out-of-5 representation discussed in Section 4.2.1 of the notes. In a pair code, each code word has exactly two 1 bits. However, one can define a pair code on any number of bits N. For example, if N=100, one has 100 bits in the code words, and exactly two 1 bits.
Your friend points out that as N grows, the fraction of valid code words drops dramatically. For N=100, for example, there are only 4950 valid code words (100 times 99 divided by 2), but there are 2^{100} bit patterns. Your friend argues that error correction capabilities for pair codes must be quite powerful, since the codes are so sparse

1. What is the Hamming distance of the pair code with 6-bit code words? Use an example to prove that your answer is correct.
2. What is the Hamming distance of the pair code on 100-bit code words? Explain how you can again prove that your answer is correct (please avoid writing 100-bit numbers).
3. How many bits can be corrected using a pair code with N-bit code words?

5. Hamming codes

The notes provided details on 7-bit Hamming codes. Consider the use of Hamming codes on more bits.

1. If a Hamming code is used with 10 bits, how many of the bits are parity check bits?
2. If a Hamming code is used with 100 bits, how many of the bits are parity check bits?
3. Why does it make little sense to use a Hamming code with 128 bits? Explain your answer.

6. SEC-DED

A communication system uses a SEC-DED (Single Error Correction-Double Error Detection) code to protect transmissions. The code is based on a 7-bit Hamming code extended with an odd parity bit (the most significant bit x_8). Using the notation from the notes, each received word consists of the parity bit followed by the 7-bit Hamming code word $x_7x_6x_5x_4x_3x_2x_1$. For each of the following received words, extract the 4-bit data word when possible, or mention that the received word had uncorrectable errors.

1. 00101101
2. 10011110
3. 10101001
4. 10010001
5. 00111001