Computer Systems Organization  
CSCI 463-1

Spring 2023

Homework #5

Formatting guide for contents of registers:

Always write the full number of hex digits (or binary) to fill out the bits of a register. In other words, a 16‑bit register must have 4 hex digits, a 12‑bit register must have 3 hex digits, and so forth. Excel will try to suppress initial zeros of a number. To prevent this, the cells for registers in this assignment have been formatted as text rather than as numbers.

Note 1: Multi-line RTL

In RTL two instructions are written on the same line when they occur during the same cycle. This occurs when there is a direct path between two components so that data does not need to go through the bus. On the datapath diagram (4b-23 and repeated elsewhere), these direct paths are shown as internal arrows. Following that convention, the last two lines of the LOAD instruction should be written as one line, i.e.:

MBR <- M[MAR], AC <- MBR

and similarly for the first two lines of STORE. Since the textbook is inconsistent about where exactly this feature is available, we will not implement this feature.

Note 2: Redundant RTL

The “execute” section of the RTL for LOAD on the reference chart below contains 3 instructions:

MAR <- X

MBR <- M[MAR]

AC <- MBR

But the “execute” section of the RTL for LOAD on slide 4c-39 only includes one instruction:

AC <- MBR

Why don’t we need the longer version? The shorter version works because the first two lines of RTL have already happened in the decode and get operand steps of the execution trace on 4c-39.

So why is the longer version used in the definition on 4b‑29 and on the reference chart below? So that you know the context in which that last line has to be executed. If the other lines weren’t there, it would look like loading any old value of MBR into AC would work.

ADD, SUBT, ADDI and JUMPI are other instructions where the first two lines of the RTL definition will already have been executed during the decode and get operand steps.

# Assignment:

1-9. Create an execution trace for lines 100-108 below. (Line 109 is HALT, which doesn’t require any instructions.)

The slides contain two complete worked examples that you can use as a guideline. One is on slide 4c-39 (repeated on 39b and 39c) and the other is on slide 4c-40 (repeated on 40b). The input to these two examples consists of the first two lines of the program on slide 4c-38.

Note that the initial values for each instruction will be the final values from the previous instruction and do not need to be repeated.

Carefully note which constants are hex and which are decimal and need to be translated to hex.

Add additional lines to the spreadsheet as needed. Leave a blank line between instructions.

If you find it simpler, you can put new information on each line in bold instead of using a background color, i.e., you can use bold text, a blue background as in the slides, or any light colored background. For your TA’s sake, please do not use a dark background color.

Note: this code contains an excerpt from a larger program and doesn’t really make any sense by itself, in case you were trying to figure out what it does.

100 LOAD Addr Initialize Next with

101 STORE Next starting address

102 LOAD Num Initialize Ctr

103 SUBT One

104 STORE Ctr

105 Loop LOAD Sum

106 ADD Num1

107 STORE Sum

108 JUMP Loop

109 End HALT

111 Addr HEX 117

112 Next HEX 0

113 Num DEC 5

114 Sum DEC 32

115 Ctr HEX 0

116 One DEC 1

117 Num1 DEC 10

118 Num2 DEC 15

119 Num3 DEC 2

11A Num4 DEC 25

11B Num5 DEC 30

10a. The RTL for ADDI (slide 4e-46) is longer than the RTL for ADD (slide 4b-29). Explain why.

10b. How many clock ticks does it take to execute ADDI in a hardwired system?

10c. Explain why.

# FAQ:

1) Fetch and decode are the same for every step.  
Fetch operand is the same for every step since MARIE is a 1-address machine.

2) If no operand needs to be fetched, the fetch operand step can be omitted.  
Lines in parentheses in the examples can be omitted, since no registers change there.

3) Only one CPU register changes on every step.

4) Final values for each step are the begining values for the next step. You don’t have to repeat the line.

5) RTL lines for the execute step that are redundant because they are already included in the decode or get operand steps can be omitted.

6) Redundant lines are inefficient but don't hurt – it’s just like saying y=x; y=x; in C++.

7) X is an abbreviation for IR[11-0].

8) You will have to hand-assemble the instructions, i.e., figure out the hex opcode and operand for each instruction. The opcodes are listed in parentheses on the MARIE summary sheet.