Department of Electrical and Computer Engineering
The University of Texas at Austin
EE 460N/382N.1, Spring 2018
Instructor: Dr. Mattan Erez
TAs: Steven Flolid, Armand Behroozi, Abraham Gonzalez

Problem Set 2 Solutions

Question 1

The following program computes the square (k*k) of a positive integer k, stored in location 0x4000 and stores the result in location 0x4002. The result is to be treated as a 16-bit unsigned number. Assumptions:

- A memory access takes 5 cycles
- The system call initiated by the HALT instruction takes 20 cycles to execute. This **does not** include the number of cycles it takes to execute the HALT instruction itself.

```
.ORIG X3000
AND R0, R0, #0          ; 9 cycles
LEA R3, NUM             ; 9 cycles
LDW R3, R3, #0          ; 15 cycles
LDW R1, R3, #0          ; 15 cycles
ADD R2, R1, #0          ; 9 cycles
LOOP  ADD R0, R0, R1    ; 9 cycles
ADD R2, R2, #-1         ; 9 cycles
BRP LOOP                ; 10 cycles for Taken / 9 for Not Taken
STW R0, R3, #1          ; 15 cycles
HALT                    ; 35 cycles
NUM   .FILL x4000
.END
```

A. How many cycles does each instruction (Do For All LC-3b Instructions) take to execute on the LC-3b microarchitecture described in Appendix C?

RTI - N/A for the problem b/c not on state diagram
ADD - 9 cycles
AND - 9 cycles

XOR - 9 cycles
TRAP
- If doing just the TRAP itself (no subroutine) then 15
- If doing subroutine 35 cycles (for HALT) or 15 + X cycles (for other)

SHF - 9 cycles
LEA - 9 cycles
LDB - 15 cycles
LDW - 15 cycles
STW - 15 cycles
STB - 15 cycles
JSR - 10 cycles
JMP - 9 cycles
BR - 10 for Taken / 9 for NotTaken

B. How many cycles does the entire program take to execute? (answer in terms of k)

To calculate the square of k, the inner loop gets executed k times. The branch is taken (k-1) times and not taken one time.
Number of cycles = 9 + 9 + 15 + 15 + 9 + (k-1)*(9 + 9 + 10) + 1*(9 + 9 + 9) + 15 + 35 = 28k + 106

C. What is the maximum value of k for which this program still works correctly? Note: Treat the input and output values as 16-bit unsigned numbers for part c. We will extend the problem to 2's complement values in part d.

k = 255

D. How will you modify this program to support negative values of k? Explain in less than 30 words.

After we load the value of k, check if it is negative. If so, take the 2's complement before entering the loop.

E. What is the new range of k (after adding support for negative values)?

k can range from -255 to +255

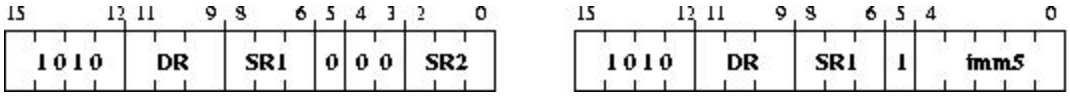
Question 2

We wish to use the unused opcode "1010" to implement a new instruction ADDM, which adds the contents of a memory location to either the contents of a register or an immediate value and stores the result into a register. The specification of this instruction is as follows:

Assembler Formats

ADDM DR, SR1, SR2 or ADDM DR, SR1, imm5

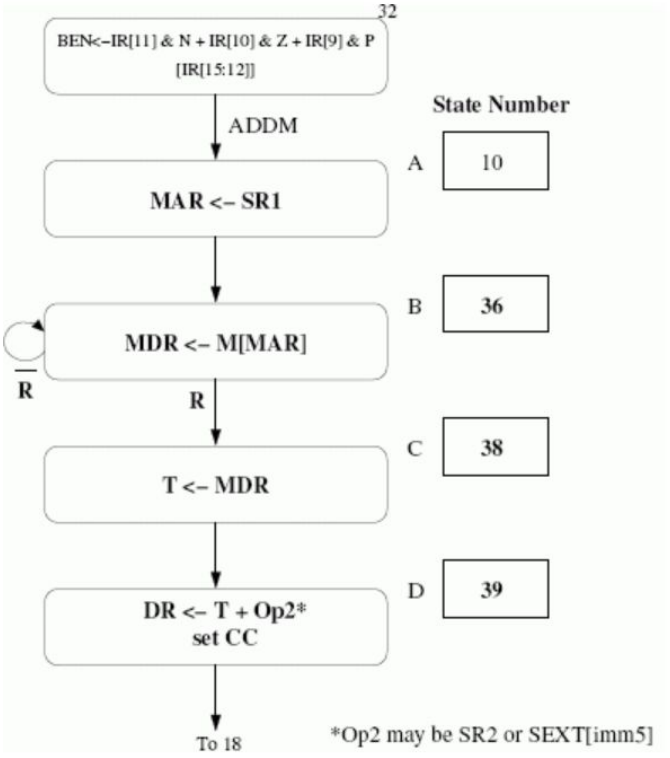
Encodings



Operation

if (bit[5] == 0) do DR = Memory[SR1] + SR2;
else do DR = Memory[SR1] + SEXT(imm5);
setcc(DR);

A. Filled in state sequence:



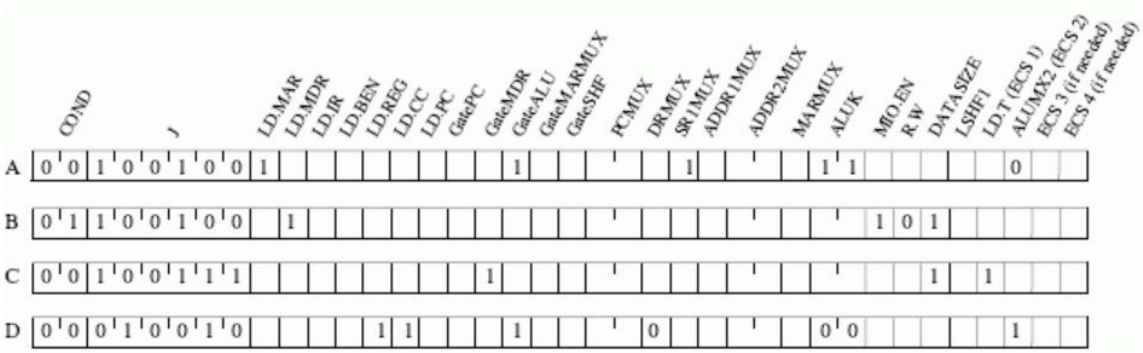
There are many possible state numberings, but state numbers must be chosen from 24, 34, and 36-63. The state number for C must differ from the state number for B only in the bit1 position, and bit1 must be 0 for state B. For example, if state B is 36 (100100), state C must be 38 (100110). The one-bit signal X is the Ready bit (R) from memory.

B. We need a 16-bit temporary register (T) which gets its inputs from the system bus. We need a signal LD.T (extra control signal 1) to control when to load this register. This register holds the data that is fetched from memory. We also need a mux in front of the A input of the ALU. This mux should select between SR1 and the output of the temporary register. We need a control signal for the select line of this mux (ALUMX2 - extra control signal 2).

   ALUMX2 = 0 selects SR1

   ALUMX2 = 1 selects T

C. Filled in microinstructions:



All other signals are 0. The J bits will depend on the state numbering chosen in part (a). The J bits for states A and B must correspond to the state number for B, the J bits for state C must correspond to the state number for D, and the J bits for D must be 18 (010010). The bit encodings for control signals are the same as specified in Lab 3.

Question 3

A. In which state(s) in the LC-3b state diagram should the LD.BEN signal be asserted? Is there a way for the LC-3b to work correctly without the LD.BEN signal? Explain.

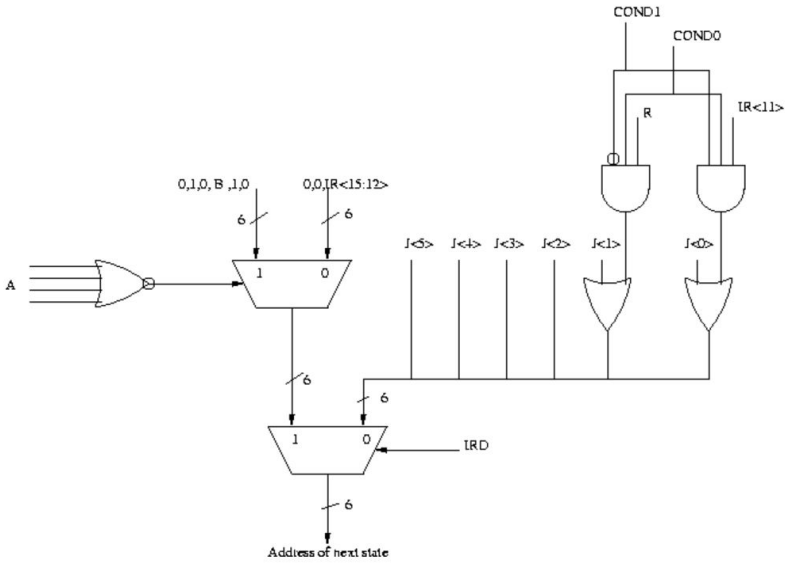
State 32. We can get rid of the LD_BEN signal altogether and always load enable the BEN register.

B. Suppose we want to get rid of the BEN register altogether. Can this be done? If so, explain how. If not, why not? Is it a good idea? Explain.

The value that is loaded into BEN in state 32 could instead be calculated in state 0, but this would add delay for calculating the next state and might cause the cycle time to be increased.

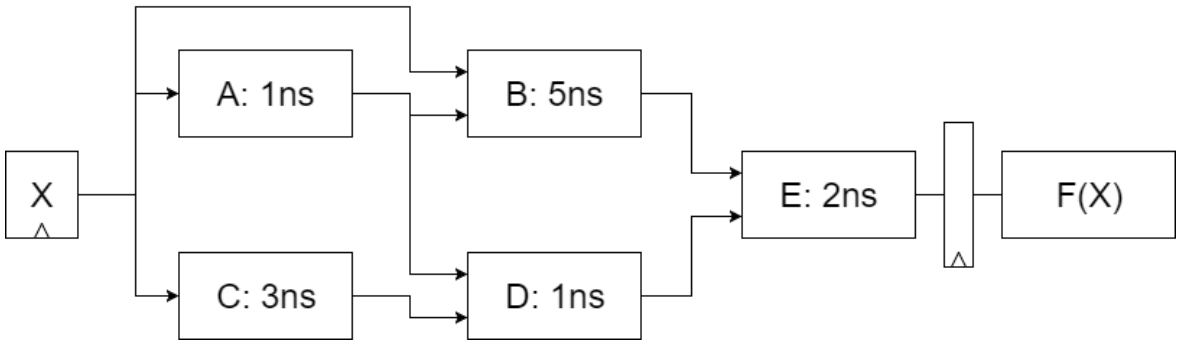
C. Suppose we took this further and wanted to get rid of state 0. We can do this by modifying the microsequencer, as shown in the figure below. What is the 4-bit signal denoted as A in the figure? What is the 1-bit signal denoted as B?

A = IR[15:12]
B = IR[11]&N + IR[10]&Z + IR[9]&P (i.e., the old BEN signal)



Question 4

Consider the circuit shown in figure below. The latency of each component is indicated in the figure and no component can be internally pipelined. Each latch adds a latency of 0.1ns.



A. How many stages does this pipeline have?

1 stage pipeline

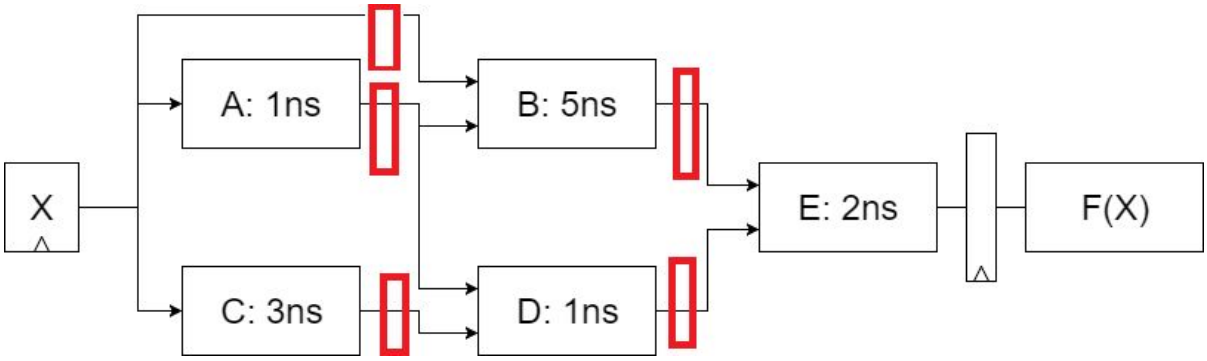
B. What is the total latency of the pipeline?

1ns + 5ns + 2.1ns = 8.1ns

C. What is the throughput of this pipeline?

1/8.1ns

D. Only add latches to the pipeline (draw your latches on the figure above OR draw below) to maximize throughput. Minimize latency per stage to maximize throughput.



E. What is the number of stages now?

3 stage pipeline

F. What is the total latency of this new pipeline?

5.1ns (1st stage) + 5.1ns (2nd stage) + 5.1ns (3rd stage) = 15.3ns

G. What is the throughput of this new pipeline?

1/(5.1ns)