CS422

Assignment 5

Simulate a lift (elevator)

Date: 7-10-2012

By

Rajat Khanduja (09010137)

Raman Anurag (09010139)

Rovin Bhandari (09010144)

Pranav Gupta (09010161)

Problem:

- Simulate a lift controller. Use the inputs to act as switches on each floor and the outputs to indicate the floor position where the lift is in, currently.

- **Priority**: Assume that you have been asked to install this lift controller in a newly set-up concern's building. Much to the dislike of the employees, the boss (of the concern) wants that the lift controller give him (i.e. his floor) the top priority. For e.g. if the boss' office is in the 4th floor then requests generated from the 4th floor will be serviced first, whatsoever. So alter your lift controller software to suit this egoistic guy.

- As time passes things take a bad turn - the boss wishes to keep a constant tab on his (lazy?) employees. He calls you up and says he wants the lift controller reprogrammed to dynamically alter the floor priority so that it is always maximum for that floor where he is currently in. So if the boss has just entered floor#5 then floor#5 has the topmost priority; later if he goes to floor#3 then it is changed to floor#3. The boss then puts down the phone when you ask him too many questions (like why the heck does he want such a priority scheme) with a threat that he would never ever place orders for your lift controller for his forthcoming buildings. Now its up to you to take this challenge or ....!

- Write the programs accordingly.

General Points about the System:

- The system allows 2 types of requests on the lift:

    1. Call the lift to a floor by pressing the button on that floor

    2. Travel to a floor in the lift by pressing floor buttons (one or more) inside the lift.

- The system does not differentiate between requests made from inside the lift or outside the lift. Both kinds of requests are made from the same input port of the LCI.

- The system provides only 1 button outside each floor. No information is monitored about the direction in which the request-making user wants to go.

- The lift uses a “Queue” to maintain the order of requests made by users. Since the size of the queue is fixed, overflowing requests may get ignored. The System also maintains an 8-bit vector at 8600H which stores the status of requests for a floor.

- The currently lit LED on the LCI output port symbolizes the lift's current position. The initial position of the lift is set to floor number 1 and the corresponding LED is shown lit.

- A polling subroutine reads the input of the LCI input port.

QUEUE

Characteristics of the Queue

- Circular Queue of fixed size.
- It provides all standard functions: ***Enqueue(), Dequeue(), IsEmpty(), IsFull().***
- Stores 16-bit numbers

Configuration in the Program:

- Memory location 8200 holds the starting position of the queue (16 bit-address)
- Memory location 8202 holds the size of the queue (16-bits)
- Memory location 8204 holds the head pointer index (16-bits)
- Memory location 8206 holds the tail pointer index (16-bits)

DESCRIPTION OF ALGORITHMS:

Part (a)

The following are also the common steps for the operation of the lift:

1. The queue is initially empty. In case of an empty queue, the system keeps on continuously calling the POLLING subroutine till a valid value from the queue is read.
2. The POLLING subroutine polls the input and enqueues it only if the bit for the corresponding floor is not set in the bit configuration vector stored at 8600H and consequently sets it.
3. The floor number of the next request is read from the queue. This value is actually a configuration vector, which has 1 for the kth bit if the request if from the kth floor. If the configuration bit is not set for this floor, the systems ignores this request and dequeues again. Otherwise, it resets the configuration bit and moves forward.

4. The source floor (current floor) and the destination floor (read from the queue) is set into register pairs HL and BC respectively and maintained throughout till the lift has reached the destination floor.

5. Also, a boolean is maintained at 8500H to store whether the destination floor is above or below the source floor.

6. To move the lift to the next floor, a small amount of delay is introduced using the DELAY subrouting to symbolize the time taken for going from one floor to another. The sign information is read from 8500H, and the current floor configuration is rotated right or left accordingly. This new value is saved in HL register pair and the corresponding LED is lit.

7. At each floor, the system checks if the bit for that floor is set in the bit configuration vector stored at 8600H and stops the lift at that floor for a certain amount of time using the DELAY subroutine.

8. Once the destination is reached, the system moves to dequeue the next request again.

Part (b) : ALGORITHM WITH BOSS FLOOR FIXED

In this part, the priority is given to the Boss only for calling the lift to his floor, but not for his movement to another floor. Algorithm is the same as the above with following exceptions:

1. We fix the boss's floor in the beginning by taking user input and store it at 8601H.

2. The request for the Boss's Floor is enqueud normally and its bit configuration set to 1 and is again reset once the Boss's request is served.

3. At each floor, the system checks if the Boss Configuration bit is set. If it is, then it does not check the configuration bit for the current floor. So the lift can not be stopped at the current floor. Further the system checks if the current floor is the Boss Floor itself. If it is, then the lift is stopped there. Otherwise the lift is moved forward to the next floor.

4. In this way, the lift does not stop on its way to the Boss's floor.

Part (c) : ALGORITHM WITH BOSS FLOOR FLEXIBLE:

This part gives the Boss priority for both kind of requests, that is, for calling the lift to his floor as well for going from his floor to another floor.

1. This part implements only 7 floors.

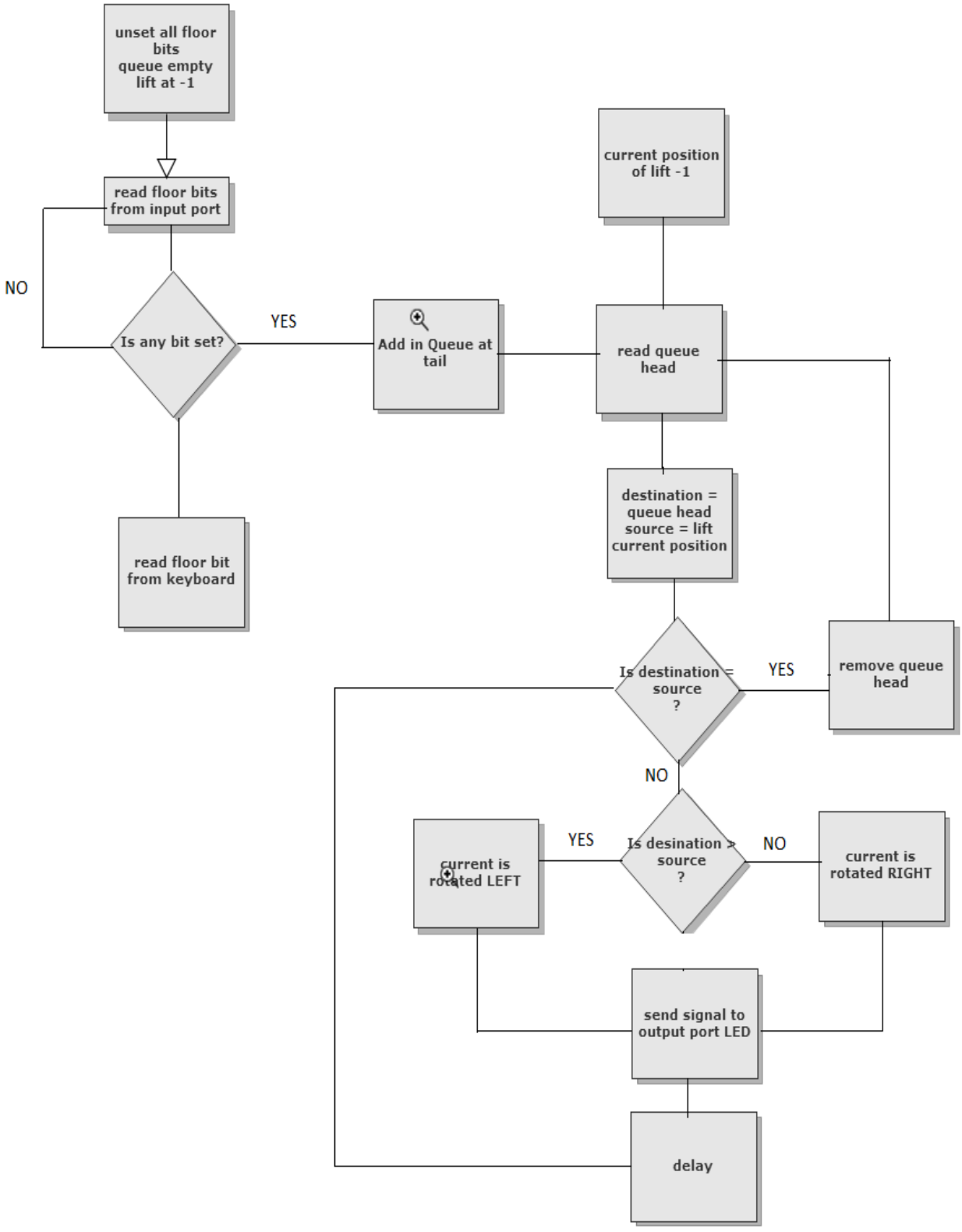
2. We provide a separate card to the boss which he uses with any of his requests. Use of the card is indicated by setting the 8^{th} bit of the input port of the LCI. Such an input ensures it is the boss making that request.

3. The boss's floor is maintained at 8601H. When a boss request occurs, his floor is set to the floor present in that request. As soon as his request is served, his floor is set to 0.

4. As in part (b), at each floor, before checking the configuration bit of that floor, the configuration bit is checked for the boss floor.

5. In earlier parts, before moving to the next floor, the system checked the sign between the source and the destination floor. But in this part, the system first checks if the Boss request is set. If it is set, then the sign between the source and the boss floor is checked and the lift is moved in the direction of the boss's floor. Otherwise the lift is moved in the direction of the destination floor. Once the boss's request is served, that is, his floor is reached, the movement towards the destination floor is restored.

6. So, in this way, if the boss's request is received, the lift turns towards the boss's floor immediately in the middle of its way itself.



Flow Diagram for executed algorithm

Assembly Code:

You can find the code for the same online as adding it in report made it very lengthy.

Link for part (a): https://github.com/pranavgupta21/CPIlab/blob/master/assignment5/lift_a.asm

Link for part (b): https://github.com/pranavgupta21/CPIlab/blob/master/assignment5/lift_b.asm

Link for part (c): https://github.com/pranavgupta21/CPIlab/blob/master/assignment5/lift_c.asm

----------END OF REPORT----------