**DLD Project**

**Deadline of submission 18th June, 2021 (Friday)**

**Project 1: Car Parking Controller**

**A Parking plaza needs to automate its car parking functionality. The parking plaza has four floors i.e. Basement, Ground Floor, First Floor and Second floor each having a capacity 10 cars. When a car comes in, the controller should tell the user about status of each floor i.e. “*Space Available*” or “*Full*”. User can choose a floor and a slot in that floor to park his car. If a floor is full, the controller will lock the entrance door of the floor and car cannot enter that floor. You have to implement the Car Parking Controller. Also your system should display the cars parked on each floor.**

***Input Signals:***

**CarIn**

**Floor to park the car**

**Slot in selected floor**

**CarOut**

***Output Signals:***

**Status of Floors**

**Door Locks**

**Parked Cars**

**Project 2: Bank Token Management System**

**A Commercial Bank needs a customer token management system. There are three counters in the bank. Counter no. 1 is for deposits, no. 2 is for withdrawals and no. 3 is for utility bills payment. Upon entering a branch, the customer goes to token machine (which displays options 1, 2 and 3 for respective counter), presses a button to get a token. If the customer has an account in the LCB, he/she can insert their ATM card at this time in the token machine. Customers with an ATM card are issued a priority token.**

**The system will display the token number for the customer being served at each counter. Whenever a customer is served, the teller presses a button, which displays the next customer’s number on the counter. If there is any priority customer, they are served first, even if there are other customers waiting before them. Assume that at any time there can be maximum 8 requests for a counter.**

***Inputs:***

**Token press button**

**ATM Card insert slot**

**Teller done serving customer button**

***Outputs:***

**Last Token number**

**Token number displays on each of the counters**

**Project 3: 4-bit Processor**

**Design a 4-bit processor which consists of 4 data registers each of 4 bits and an instruction register (IR) of 7 bits. The first 3 bits of the instruction tells which operation is to be performed, the next 2 bits signifies the first register and the last two bits signifies the second register.**

|  |  |  |
| --- | --- | --- |
| **I6-I4** | **I3-I2** | **I1 – I0** |
| **Operation Code** | **4-bit register operand 1(R1)** | **4-bit register operand 2 (R2)** |

**The following operations are performed by the processor.**

|  |  |  |
| --- | --- | --- |
| **Operation Code** | **Operation Performed** | **Description** |
| **000** | **R1 = A** | **Load the contents of input A in to the register operand 1.** |
| **001** | **R1 = R2** | **Move the contents of register operand 2 in to register operand 1.** |
| **010** | **R1 = R1 + R2** | **Add the contents of register operand 1 and register operand 2 and load in register operand 1.** |
| **011** | **R1 = R1 - R2** | **Subtract the contents of register operand 2 from register operand 1 and load in register operand 1.** |
| **100** | **R1 = R1 \* R2** | **Multiply the contents of register operand 1 and register operand 2 and load in register operand 1 and 2. (As the result is of 8 bits)** |
| **101** | **R1 = R1/2i** | **Divide the register contents of register operand 1 with 2i (i is an input) and load the result in register operand 1.** |
| **110** | **R1 = R1\*2I** | **Multiply the register contents of register operand 1 with 2i (i is an input) and load the result in register operand 1.** |
| **111** | **R1 = R1 + R2** | **Logical OR the contents of register operand 1 and register operand 2 and load in register operand 1.** |

***Inputs:***

**Clock Pulse (CP), 7-bits Instruction, A, i**

***Output:***

**Contents of each register**

**Project 4: Snakes and Ladders**

**It’s a two player game. You have 0-31 cells, with each cell having a unique number. Each player roles a dice and output can only be between 1 to 6. The player moves number of cells ahead according to face value of dice. The first one to reach 31 will win. You need to fix the positions of snakes and ladders on the cells. The game also has to display the turn of each player.**

***Input***

**Clock Pulse CP**

**Start Dice Roll 1 to start and 0 to stop**

***Output***

**Position of each player on board**

**Winner Player WP**

**Note: There should be at least 3 snakes and 2 ladders.**

**Project 5 – Modified Stack**

**A stack is a Last In First Out data structure that saves the elements in an array such that elements are added and removed from one end only (i.e. top). ModifiedStack is a special type of stack which saves latest MAXSIZE elements and returns them one by one.**

**For example, if MAXSIZE is 4 this means that ModifiedStack will remember the latest four elements --- it will remember the first four integers saved ‘2’, ‘4’, ‘6’ and ‘8’ and if the fifth integer ‘10’ is saved it will only remember ‘4’, ‘6’, ‘8’ and ‘10’. The ModifiedStack will now return ‘10’, ‘8’, ‘6’ and ‘4’, one by one, when needed.**

**Functionality Required:**

**- Storage of 10 integers ranging from (0)10 to (15)10**

**- Push element in ModifiedStack**

**- Pop element from ModifiedStack, if Stack is empty then inform the user**

**- User friendly interface**

**Project 6 - Queue**

**A queue is a First In First Out data structure that saves the elements in an array such that elements are added from one end (head) and removed from the other end (tail). For example, if user saves elements ‘2’, ‘4’, ‘6’, ‘8’ one by one the queue will return ‘2’, ‘4’, ‘6’ and ‘8’ one by one when required.**

**Functionality Required:**

**- Storage of 10 integers ranging from (0)10 to (15)10**

**- Insert element in queue, if queue is already full then inform the user**

**- Remove element from queue, if queue is empty then inform the user**

**- User friendly interface**

**Project 7 - Digital Smart Lock Room**

**We have to automate six lights and a lock of a room. The room has six enclosures and requirements are as follows:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Light 6   |  | | --- | | S15 | | S14 | | Light 5   |  |  | | --- | --- | | S11 |  | |  | S8 | | Light 4   |  | | --- | |  | | S12 | |
| Light 3   |  | | --- | |  | | S6 | | Light 2   |  |  | | --- | --- | |  | S4 | | S3 | S2 | | Light 1   |  | | --- | | S1 | | S0 | |

1. **Capacity of room is 16 people**
2. **Seat plan is shown above and user entering the room will sit on next available seat starting from seat no. 1**
3. **Only person at seat 1 can exit the room and on his exit all the people will move to previous seat**
4. **The room should be able to display the number of people inside**
5. **If all the seats in an enclosure are vacant then its light will be turned off**
6. **If the room is full its Entry door will be locked and the exit door will be locked if there is no one in the room**