University of Toronto, Faculty of Applied Science and Engineering

Department of Electrical and Computer Engineering

**ECE 243S – Computer Organization – 2015**

**Project Proposal Form**

After ensuring that your project idea is unique, you will use this form to describe your project (point form preferred), assess its difficulty, and outline what you expect to achieve each week of your project work. You **must submit the filled form on Blackboard two days before your scheduled project lab session (March 15-17) and bring it to the first project lab session (March 17-19).**

The TA will advise you if changes are needed to your project proposal so it is sufficiently, but not overly challenging. After you implement the changes, **the TA will then approve your project proposal.** You will then make **two copies** of the final filled form: one will be kept by the TA, and the other one will be for your reference. Your ability to successfully implement all that was approved in your proposal will determine your project functionality marks.

# Group Info

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Station Number** | **First Name** | **Last Name** | **Student Number** | **Contribution [0..100]**  **(filled during 3rd lab)** |
| 20 | Gokul Kumar | Kaushik | 999878191 |  |
| 20 | Hengyue | Chen | 1000011667 |  |

# One Sentence Project Description (as posted on Piazza)

|  |
| --- |
| Project Idea: Assembly Implementation of 2048 game with VGA.  Actual Game: <http://gabrielecirulli.github.io/2048/> |

# Technical Description of the Project

Describe your project in more technical details and include a system block diagram.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **About the Game:**  2048 is a game where you move blocks containing values of powers of two (i.e. 2,4,8,16,32…). You can join blocks of equal value, move blocks into empty adjacent spaces or pile them up. Joining blocks of equal value is called merging. Moving blocks into empty adjacent spaces is called shifting. The movement can be done by pressing one of the four movement keys of the keyboard : ie. Up, down, left right and ALL the possible block movements will be made in that direction.  Win:  The goal (win condition of the game) is to merge blocks until you get to 2048. Another goal (ideal/sub-goal) would be to finish the game with the least number of moves.  Loose:  The loose condition of the game can be from any of the following:   1. Running out of time 2. Running out of space to move blocks (no logical moves) 3. Giving up (by quitting)   In the online version, the time aspect does not exist. It has been added as a user challenge and to double up as a 2nd meaningful interrupt.  Note: Thought the online version of the game functions with animation, the assembly implementation WILL NOT.  A picture of the original game:    **Code Visualization:**  The project is basically playing the game of 2048. There are multiple aspects of it. The code structure will use the following pattern:   1. Set up the interrupts of TIMER and JTAG UART [escape key only] which are the games loosing criterion (if the TIMER finishes its time or the ESC key is played, the game will end). 2. Poll JTAG Uart for the movement direction 3. The “2048 algorithm” itself which rewrites the array after every move. 4. The number of moves made will be written to the HEX display function. 5. The C function is called which will return a “random” spot where the location of the next block and its block value (i.e. 2,4,8) is given. 6. The “Write to VGA” function displaying current block positions on screen. (This can be a C function or assembly depending on what fetches more marks) 7. The “Write to other outputs” which writes to the proposed LEDs and 16\*2 LCD screen. 8. The “Play audio codec” function that makes some movement noise after every play. (Can be C function also) 9. The game repeats this loop. 10. The interrupts of the timer will update till a certain point (i.e. the game end) and the TIMER will interrupt every (say 1 minute) and then the interrupt will take place and the LED display will be updated (i.e. how much time left based on a division by 16).   **Inputs and Outputs:**   |  |  | | --- | --- | | **Inputs** | **Outputs** | | JTAG UART Polling | VGA Display | | JTAG UART [Interrupt] | LED Display (16, RED) | | TIMER [Interrupt] | LCD Display | |  | HEX Display | |  | Audio Codec |   **Project Setup: (System Block Diagram) BLUE = OUTPUT; GREEN = INPUT**    The project setup shows how things are connected but don’t specify which component shows a given value:   |  |  |  |  | | --- | --- | --- | --- | | **Input** | **What it Controls** | **Output** | **What it represents** | | Keyboard (JTAG UART) | Direction Movement (up, down, left, right) | VGA Display | Shows the blocks in their current state (redrawn w/ NO animation) | | Keyboard (JTAG UART) interrupt | Quitting the game (ESC) | HEX Display (7seg) | Displays No. of moves taken by the player | | TIMER interrupt | How much time left to loose the game | LEDs (RED) | Show how much time filling up the LED spots leaves as the time goes up. | |  |  | Audio Codec | Plays the game theme and makes “keyboard noise” indicating movements | |

# Technical Description of the Project Core

The project core is a minimum part of your project that you are committing to deliver. Failing to implement this part will result in loss of functionality marks.

Describe your project core here.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The project core is essentially implementing the following aspects stated in the project idea section.  The project core inputs and outputs are here. (The one’s **not** part of the core highlighted with red) :  **Inputs and Outputs:**   |  |  | | --- | --- | | **Inputs** | **Outputs** | | JTAG UART Polling | VGA Display | | JTAG UART [Interrupt] | LED Display (16, RED) | | TIMER [Interrupt] | LCD Display | |  | 7seg Display | |  | Audio Codec |   Others:  Linking C with Assembly is also part of the project core.  The project core diagram is as follows: |

# Assessment of Project’s Difficulty

Please check off each accomplishment you propose in your project and indicate whether that accomplishment is part of the project core and whether it is interrupt-driven (if applicable). For accomplishments with multiple units such as the LEDs, switches, motors, etc., indicate the number of such units used. For example if you are using two Lego motors place the number 2 in the column instead of a checkmark.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Accomplishment** | **Proposed?** | **Project Core?** | **Interrupt?** | **Demonstrated?**  **(to be filled by your TA)** |
| LEDs/Switches | YES |  | N/A |  |
| Push buttons |  |  |  |  |
| Digital protoboard |  |  |  |  |
| VGA | YES | YES | N/A |  |
| 16x2 LCD | YES |  | N/A |  |
| Lego motors |  |  | N/A |  |
| Lego sensors |  |  |  |  |
| Linking C with assembly | YES | YES | N/A |  |
| JTAG UART transmit | YES |  |  |  |
| JTAG UART receive | YES | YES | YES |  |
| Timer | YES | YES | YES |  |
| Hexkeypad (rows or columns only) |  |  |  |  |
| Hexkeypad (rows and columns) |  |  |  |  |
| RS-232 UART transmit |  |  |  |  |
| RS-232 UART receive |  |  |  |  |
| DMA transfer |  |  |  |  |
| Nios II Custom Instruction |  |  | N/A |  |
| Audio Codec output to speakers | YES | YES |  |  |
| Audio Codec input from microphone |  |  |  |  |
| PS/2 Keyboard |  |  |  |  |
| PS/2 Mouse |  |  |  |  |
| SD Card Reader |  |  |  |  |
| Custom Bus Component |  |  |  |  |
| Ethernet |  |  |  |  |
| 7seg Display | YES |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Please describe any other devices or complex software algorithms you will use. Remember to keep this relevant to ECE243 (not fancy electronic circuits or complex mechanical systems).

|  |
| --- |
| Complex Algorithm self developed to make 2048 block movements that is tailored specifically to the NIOS-II assembly language. |

# Project Milestones

Describe what parts of your project you will have fully implemented in each of the three project lab sessions. Keep in mind that you will have to demonstrate your project during the third project lab session. The key here is to design incrementally: get something working quickly and keep adding to it. TAs will not accept the “integrate everything in week 3” approach.

## First project lab session (week of March 17)

|  |
| --- |
| **Implement 2048 Algorithm**  **Implement Polling of JTAG UART**  **Implement Interrupt of JTAG UART**  Implement TIMER Interrupt  Show some basic knowledge of VGA output capability  Figure out how to get the audio codec running |

## Second project lab session (week of March 31)

|  |
| --- |
| #Decide after meeting the TAs.  Implement Victory and Loss conditions  Implement Screen changes  /\*  Tentatively:  Implement Graphic VGA output  Get the audio codec running.  \*/ |

## Third project lab session (week of April 7) – Demo

|  |
| --- |
| #Everything. |

# TA Notes

This page is filled by your TA.

## Approval

Approved by \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## First project lab session (week of March 17)

|  |
| --- |
|  |

## Second project lab session (week of March 31)

|  |
| --- |
|  |

## Third project lab session (the week of April 7) – Demo

|  |
| --- |
|  |